BACHELOR THESIS

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

BACHELOR IN COMPUTER SCIENCE AND ENGINEERING

DESIGN AND IMPLEMENTATION OF A GENETIC ALGORITHM-BASED SYSTEM FOR MANAGEMENT AND OPTIMIZATION OF SOFTWARE DEVELOPMENT WORKING TEAMS

Conducted by:

Elena Hernández Sánchez

Supervised by:

D. Juan Miguel Gómez Berbis

D. Enrique Jiménez Domingo

Acknowledgements

To all the people who has been by my side along this journey, to family, friends, teachers but most of all to Antonio, which has been the greatest support along the bachelor degree and during the development of this project.

Abstract

This project combines together topics and technologies related to human resources, team management and software development methodologies all combined with the support of Genetic Algorithms.

The aim of this project is the optimization and management of software development working teams; its purpose is to help people in charge of managing teams at the time of configuring and designing working teams, saving them time, money and helping them to design the most suitable team for a project.

In order to achieve this objective, a Genetic Algorithm-Based System was designed. The objective of the system is the creation and selection of the fittest team configurations given a certain set of parameters specific to the project.

The results of the experimentation process demonstrated that the system is capable of providing desirable team configurations which match in a significant percentage with the teams that an expert project manager would configure.

This project could make a difference in the way working teams are traditionally configured and there could be a new path to follow by extending it to different areas of work and study.

Keywords:

Team, Individual, Capability, Skill, Software Engineering, Methodology, Artificial Intelligence, Genetic Algorithms

Table of Contents

List of Figures

List of Tables

List of Equations

PART I:

INTRODUCTION AND OBJECTIVES

Chapter 1 | Introduction

The project "*Design and Implementation of a Genetic Algorithm-Based System for management and optimization of Software Development Working Teams*" was born with the objective of helping the companies at the time of configuring and optimizing software development working teams, and thereby helping also the team members by including them in a group more consistent and complementary with their particular abilities than the ones they were in.

1.1. Description of the study fields

The final goal of this project is the optimization of software development teams. In order to achieve this goal, several fields of study have to be approached and studied in detail so as to have the highest level of knowledge possible on each subject and find the most suitable ways to combine different elements together.

The traditional way of managing working team is no longer valid, now it is not enough to put together a team of people with similar knowledge in the same fields and expect that group to work together harmoniously obtaining the best results and having the better productivity.

Team work has changed, as the way people communicates and works has also changed with it, now working teams do not have to have the same sort of people to be the best team, know, variety and heterogeneity may provide the key to the true success of a project, if used wisely; just because your team is heterogeneous does not imply that it will automatically success.

Nowadays, the human factor, the individual himself, is much more important as before, individualism has raised as well as the development of individual capabilities; people have more trouble working with others, and a significant amount of workers would rather work alone in their projects than having to deal with others and the different obstacles and setbacks that communication and work with others may imply.

Even though individualism has raised, and people are more solitary, teamwork spirit should be encouraged more than ever; whit the wide range of existing technologies and available knowledge, it is very unlikely that a person is an expert in all the different aspects which are important to successfully develop a particular project; people need

each other, to complement their knowledge; to compensate their lack of skills on one subject with the expertise of his team mate in such topic.

Teamwork is about sharing; about contributing to the team with everything the individual has to offer, and making it common knowledge for the greater good of the team.

Configuring a working team properly, could save companies a lot of time and of course, a lot of money; working teams which are blended, which work together as a whole, are able of providing better results, and to provide them sooner, because they communicate and help each other, they don't believe in individualism and the maximum goal is not the success of the individual, but the success of the whole team together.

It is important that project managers, and consequently the companies, take some time to think about how they want to configure the team which is going to be in charge of the development of a project, it is not a decision which should be taken lightly and, "wasting" some more time looking at the more specific characteristics of each worker composing the team, could tour that "waste" into profit for the manager, the company, and of course, the workers which were put together in the team.

Motivated and happy workers usually work better, with more enthusiasm and faster, all these factors are considered profitable for the company, and they could be achieved by only devote a significant small portion of time before the actual development of the project, to think which will be the perfect team for that project.

1.2. Solution approach

As it has been stated before, devoting time to the configuration of the perfect team should be one of the priorities of project managers before actually deciding which people are going to be on their team but, what normally happens is that this small portion of time is not found in the project manager's schedule and what usually happens is that development teams are formed without taking into consideration the human factor, without putting all the individuals together and really thinking if they individually and as a whole, configure the optimal team.

Team management is an important issue to which organizations and managers should pay attention to, the success or the failure of a project may depend on the proper configuration of the workers configuring the team in charge of the development of such project.

This project has been studied and designed with the objective of making life easier for companies, project managers and everyone who may benefit by the use of this management and optimization tool. It is known that managers do not usually have time to do all the human resources related tasks as possible, and maybe they do not have knowledge enough on the subject to do it.

Sometimes, what could happens, is that the person in charge of defining the team does not know well enough the workers he can count on, and ends up teaming up the workers he knows better, regardless if this would be the best possible combination from the available set of workers, because the others are unknown to him.

With the help of this system, the individual capabilities, skills and competence of each different worker available for the current project's team configuration would be registered; the evaluation of each worker could be defined by taking auto evaluation test or by doing 360º reviews between workers which already know each other. Depending on the evolution of the worker, their personal and professional skills could be easily updated.

Thanks to the assistance of Genetic Algorithms, this tool would be capable of aiding project managers and companies in the difficult and heavy tasks of configuring the most suitable working team for the desired project.

This system would help the person in charge of configuring the team by suggesting him the best or the best team combinations it considers the bests ones with the given set of available workers and the parameters set for the project. Furthermore, if the project is focused in an specific area of development, the system gives the user the possibility to find the best team at some capability, and for example, design the team with the best programming skills for a particular project.

In case the project should be developed relying on some methodology, the system offers the companies the possibility of looking for the most suitable individuals for the defined

project, according to the principles and needs of the methodology which has been set by the user.

Summarizing, the purpose of this project is encouraging smart team building, taking into account personal and professional aspects of the workers in it, and giving the best configuration obtained to the user, making his decisions easier at the time of designing a software development working team and helping him and the team to have a better work environment, increase productivity and reduce development conts.

1.3. Report structure

This document is divided in seven different Parts; each Part contains one or more chapters which at the same time consist on multiple sections and subsections.

The first Part is called “Introduction and Objectives” and it is composed of two chapters (1 and 2). In the first chapter, a general introduction to the subject of the project, the solution to the initial problem and the structure of this document are contained. The second chapter discusses about the objective of this project and the motivation which encouraged me to investigate on this theme.

The second Part, “State of the Art” contains four chapters (3, 4, 5 and 6) and it provides a general glance at the current situations and the latest advances on all the different fields of study which are analyzed and used in this project. The first of these chapters verses about the working teams, how they are composed, which factors determine the success of a team and so on; in chapter 4, the Human Factor is thoroughly analyzed, the impact of a person’s attribute on the development of a project, the relationship with his team mates, and the most desirable competences a user should have. In the fifth chapter different Software Development Methodologies and Frameworks are analyzed, going from Traditional methodologies to Agile ones and extracting conclusions on the human factors which are most important for those methodologies. In chapter 6, the last of this Part, the main tool which is going to be used for the development of this project will be described and studied, which are the Genetic Algorithms, their history, their modalities and how they can help solving real file problems.

The third part is named “Analysis”, it is divided in two chapters (7 and 8) and it contains all the required information about the prior analysis of the System needs and requirements. Chapter 7 provides the reader with a Conceptual Model of the most important data the proposed System is going to be handling and managing, and in chapter 8 all the different requirements of the System are going to be stated and defined, as well as the different Use Cases scenarios in which the user will be involved with the System.

The fourth part of this document is the “Architecture” and it has two chapters (9 and 10). In chapter 9 the general structure of the System Architecture is defined and explained, and in chapter 10 all the different Classes which compose the System are listed and described in detail for better understanding of the functioning of the System.

The fifth part, “Description of the Solution” is composed of three chapters (11, 12 and 13) and it provides an extensive and detailed description of the proposed problem, and its solution. In chapter 11 the whole solution is explained from the beginning, the justification of the choices made and the entire process to follow. The next chapter contains an example case study in which the solution provided is tested and checked and the final chapter of this part contains several designs representing the Graphical User Interface of the final system.

The sixth part, “Conclusions and Future Lines of Work” only contains a chapter (14) and it offers the reader personal conclusions on the work done after the conclusion of the project, difficulties encountered along the process and future lines of work to be developed and studied starting from this project.

Finally, the seventh Part, “Appendices” contains 5 appendices, each of them containing different information besides the actual project such as definitions, planning and budgeting, tools used in the development, and the list of references used along the whole development and documentation stages of the project.

Chapter 2 | Objectives and Motivation

In this chapter, the main objectives which must be met in this project will be stated and explained, as well as the major set of motivations which lead me to the development of this project.

2.1. Objectives

This project was born with the objective of providing and effective and efficient way of optimizing working teams' configuration, so as to help the companies to increase their productivity and save money by using this system.

In order to reach this main objective, there are a series of intermediate objectives which have to be met before:

- Study and know the actual situation of working teams in the software development companies.
- Define the importance of the human factors in project development and which of these competences are more important at the time of configuring a team.
- Establish a technology which will be the tool used to process and refine the entry data.
- Design a system which is capable of handling the problem and providing satisfactory results.
- Validate and test the system with examples and study cases.

Besides these main objectives which are directly related to the main goal, it is also important that the designed system is easy to use for any kinds of user, not only computer scientist and it is also important that it has a friendly and intuitive user interface.

2.2. Motivation

Working teams are always present, either in college, at work… We are continuously required to work in teams with other people for different kinds of projects. Depending on the team one is into, the final results could be better or worse and in some occasions, catastrophic.

The importance given to the proper configuration of working teams is still very little, and working teams are configured almost randomly, with no defined criteria to put one worker into a team and not some other.

Because of this little concern to working teams' configuration, many different problems may arise if the group has not been configured properly, team members may not get along, the lack of technical capabilities may arise at critical moments, and the project development could be drastically harmed if the team developing it is not working as a whole.

The motivation to develop this project comes from this, the thought that companies should take a little more time at the time of configuring working teams, because the benefits of having a well configured, balanced and optimized team are countless in comparison with the time spent designing the team members.

In small companies, this process can be done manually, with the help of human resources personnel and project managers, but in big companies it would be very costly to personally evaluate all the workers and manually configure the teams.

For these reasons arises the idea and the motivation of designing a program which is able to help companies at the time of designing working teams, save them time, money and provide them better productivity and results.

PART II:

STATE OF THE ART

Chapter 3 | Working teams

In this chapter, the concept of team will be analyzed, and it will be extended to the field of Software Development,

3.1. Teams

According to the Oxford Dictionary, a team is "*Two or more people working together*"; people are social animals and therefore the natural tendency is to collaborate and work with each other for a common purpose.

It is important to make a distinction between groups and teams; in the company environment, a group represents a set of workers with similar knowledge working in the same area where each of the workers has their own tasks to perform individually from the other workers, there may be communication and related tasks between them, but at the end they do not have to work together in order to fulfill their tasks.

On the other hand, a team implies that that set of workers have to work together, as a whole, they have to have common goals, objectives and have a similar way of thinking in order to be a successful team. Teams are necessary because in many occasions carrying a whole set of tasks individually would not be possible without the help and support of the team mates.

In a team, their members tend to have complementary skills, and the lacks or weaknesses of someone are usually compensated by the strengths of his partner, and so on. Teams are about collaboration; team members should help each other and create a relaxed and comfortable environment in which all the workers are happy to be in the team and looking forward to get things done.

Unlike work groups, in which members are assigned to the same area and perform similar tasks, in teams, their members do not have belong to the same area, or have the same level of knowledge on some subjects.

In working teams, there are different levels of responsibility and different areas of expertise, and each team member has to be able to fulfill their task in a way that it is consistent and with the work of the other team members, so communication is a key factor in working teams which could determine the success or the failure of a project.

3.2. Software Development Teams and Roles

Software development implies the process of developing a software product so; a software development team is a team whose objective is the development of a software product.

Depending on the type of project, and on the working methodology applied, there are several roles in which different team members can be classified inside a development teams and they widely vary depending on several factors; but there are some roles which are present more of the times and are the ones which are going to be considered in this chapter.

3.3. Project Manager

The project manager exists in any kind of projects, and he is the person in charge of managing everything, of making sure that the project is a success and the objectives defined are reached. He has to take care of the workers which are going to be part of his project, as well as managing the resources available and allocated for the project.

Other of the main responsibilities of the project manager is managing time, he has to make sure that deadlines are met in time, and that everything is being developed as expected following the planning.

The project manager has to be the agent of the client inside the development team, it is his job to make sure that the client's requirements are being followed and fulfilled as expected, and that the client is going to be happy when he receives the final product because it will be exactly what he wanted. For this reason, the project manager has to have business and technical knowledge, to be the intermediary between the client and the developers.

3.4. System Architect

The system architect is the person in charge of defining and designing the basic structure of the system which is going to be designed. They are in charge of materializing all the client's requirements in terms of the system.

It is necessary that the system architect is able of defining the system architecture along with its subsystems, which should be consistent with further development and should be unlikely to be changed; they have to take the client's vision and transform it into defined and specific requirements which will define the system functionalities and constraints; and this requirements will be followed in the development phase to construct what the architect has previously designed.

System architects are required to communicate with the clients in order to fully understand their requirements and needs which will be constantly changing. They have also to define which will be the best way to implement the whole system definition by analyzing the costs that different implementations would imply.

3.5. Development Manager

The development manager is the person responsible of all the different operations which are performed every day, he responds to the project manager, and his main priority is to deliver a product, to transform the pieces of development into a functional and complete product which could be sold to a client.

He is in charge of almost everything related with the development of the project, he has to be able to define the scope of the project, to design, prioritize and assign tasks to the other developers, as well as estimating how much these tasks are going to require in terms of time and resources, being a guide and a counselor to the other developers and taking them in the right direction.

The development manager has to make sure that all the developers' work is coordinated and that the work-flow is being properly followed. He is in charge of the communication with the whole development team and to make sure that everything is being developed as it should in the more efficient manner possible.

Additionally, the development manager is the person standing between the project manager and the development team, and he has to represent the interests and opinions of the developers and transmit them to the project manager.

3.6. Developer

A software developer is the person whose work implies all the different stages which compose the development of a software project which usually implies planning, design, implementation and testing.

They must be able to contribute with their work at all the different stages of the development process and be involved in every of them and they will respond to the development manager following the designs and requirements definitions of the system architect.

A developer is in charge of following the development process from the beginning, from making consistent design of the program, producing quality code and testing every important aspect of the product to be developed to assure maximum quality. Of course, developers have to be aware of the planning and deliveries scheduled and will have to follow them religiously.

3.7. Business Analyst

A business analyst is a person with both technical knowledge and has the business vision which helps him to understand the client's wishes. He has to be the link between the user and the system architect; he must be able to translate what the client wants in business terms, into technical and concise language so it will be easier to the architect to represent the client's requirements in his designs.

This role was mainly created due to the communication problems existing between the clients and the technicians, in order to avoid miscommunication between these two parties and to speed the process of requirements definition and make these requirements more accurate and accord to the client's desires.

Chapter 4 | The Human Factor

This chapter will depict and describe the importance of the human factor in software development teams, as well as some capabilities which have been considered important at the time of configuring a team based on the abilities of their components.

4.1. Personal capabilities

In this section, a set of competences and capabilities related to the person will be described. These abilities are strictly related to the person, and most of them cannot be learnt like programming; they are inherent to the worker and they are what makes him unique and different from the others.

Even though personal capabilities cannot be strictly learnt, they can be improved, and negative behavior can be corrected if the person is whiling enough. As well, if someone is naturally good as something, he can exploit this capability to make him stand out, and leave his weaknesses in the background.

4.1.1. Teamwork

It is known that, generally, people need the supports of the other at the time of making critical decisions, the guidance and the advice of the others is often essential at the time of making a decision or having a great idea.

The motto of all members of a team must be “There is no I in Team Work”, that is the main idea after the teamwork capability, the team must come first for every work aspect of the team members

Teamwork is a critical competence which could determine the success of the work of a team; workers have to be able to collaborate with each other, to discuss the aspects in which they disagree without criticizing the others’ work and to propose alternative solutions when they don’t like the current situation.

Communication is also a key element which can make teamwork success, team members have to be able to effectively communicate with each other to avoid misunderstandings, to be transparent and to have no fear to express their thoughts to the rest of the team.

Of course, healthy competition among team members and auto evaluation is encouraged as long as it is used as a means of motivating team members to achieve their objectives with the best performance possible. The members which perform best and give their best for the team should be rewarded and recognized.

4.1.2. Leadership

A leader is a person that is in charge of others, of influencing the group towards a goal, by guiding and managing them.

A leader is necessary in a team because it will be the one managing all the workers, motivating them to keep the focus, helping them at the time of making decisions, and always listening to what every team member has to say, helping the team to make decisions and set goals together, because they are a team above all.

A team leader must be someone impartial, who evaluates the workers equally, for the performance they are providing to the whole team in order to avoid further problems with partiality and favoritism. The leader must be the one in charge of sticking the team together, to encourage everyone to collaborate and to trust each other with their doubts.

Leadership also implies teaching and caring; the leader must be able to mentor the workers, help them with their doubts and stand by them when there are problems, assuming the responsibility of everything that happens in his team.

If there is one thing to keep in mind, is that leadership is a two-edged capability, because all teams must have a leader, but it is not recommended that a team has more than one leader, because they may collide and cause more bad than good to the whole team stability.

Leaders are usually strong people, which are used to the fact that people listen to them, and follow their lead, this is desirable for a team without a leader, because the workers need guidance and direction; but when the team already has a leader, these two strong personalities are likely to crash in case one of them does not bow to the leadership of the other.

4.1.3. Commitment

According Oxford Dictionary, commitment is "*the state or quality of being dedicated to a cause, activity, etc.*",

Commitments is a critical factor for the success of a team, if team members are dedicated to the team, to the project, they will be serious at the time of getting things done, and devote the necessary time and resources to achieve a goal.

There are three different and complementary levels in which the commitment of a person, regarding a company, can be included:

- **Commitment to the team members**: The person is fully dedicated to the rest of the team members, and he gives his support and care to the others in other to have a successful team in which everybody matters. With this type of commitment, stress level is lowered and duties are more easily fulfilled as workers know they have their peers to support them.

- **Commitment to the project**: the worker is fully dedicated to the project, he understands his role in the team and its ultimate goal is to make the project succeed. Team members will put all their effort in the successful development of the project and will work together to make it happen.

- **Commitment to the company**: This third level of commitment implies that the worker is fully devoted to the company he is working in, and its objective is that the whole organization manages to reach its goals by providing all he can from his work in order to contribute to the global success of the company.

For this particular project, the most relevant types of commitments are the first two, to the team and to the project. It is important that team members acknowledge the importance of the project they are on and put all of their effort and dedication into it, as well as trying hard to be the best partner and help those in his team. The stronger these commitments are in the different team members, the greater the chances of success for the team, the project and the company.

4.1.4. Planning

Organizational skills are an important and valuable attribute at the time of working in teams; team members have to be able to organize themselves, and then to organize with

the whole team. For a worker to be organized, he has to do accurate planning and follow it, optimize the available time, prioritize tasks and detail orientation among others.

- **Planning**: is the ability of thinking ahead, to define when something is going to be started and when is going to be completed, and sticking to that planning without changing it. It also implies planning on advance which set of tasks are going to be required in order to achieve a goal.

- **Optimize time**: is the ability of managing time, to make the most of the available time and to use it wisely. In order to optimize time, it is necessary to properly design an schedule, and follow it, use it to include any changes on the planning or important due dates to take them into account.

- **Prioritize tasks**: is the ability of knowing which tasks are more important than the orders, and once that is defined, wisely distributed the available time and resources according to that priority list in order to complete the critical tasks first and improve productivity.

- **Detail Orientation**: is the ability of paying attention to details, to the small things which could imply the big changes. It is important to keep an eye on everything, and make sure that every aspect is completed when a task is finished in order to avoid further problems because of missing details that could seem meaningless at the time.

4.1.5. Creativity

Creativity is "*the use of imagination or original ideas to create something*". In software development creativity has a significant importance because when developing software, something new is created, and it has to be original, and different from previous systems; the key to getting unique software is using creativity to capture original and innovative ideas and transform it into something great.

Creative people must be whiling to think outside the box, to search for something that is needed but has never been created, to do something simple, but in a different way to make it stand out; they have a unique vision of the world that not many people has.

In software development, creativity is important at the time of defining new concepts and ideas, but it also important at the time of giving shape to these ideas, like for

example, designing the graphic interface of an application, a creative person will be able to get the most of what is available and give it a desirable look which may be emulated by others in the future.

A creative person must be able to speak out their thoughts and to share their ideas with the team workers, to listen to people opinions to actually understand what someone wants from something, what is expected and what could be improved; they should not be afraid of experimentation, because after several trials and errors great ideas may arise.

Besides, creative people must be encouraged by their teams and their bosses to exploit that creativity because, if no means are provided and no freedom is granted, all the creativity and the ideas that a worker may offer to the project, will remain confined without the possibility of taking advantage of it.

4.1.6. Social Skills

Social or skills, or the ability to effectively communicate and interact with others, is important in every aspect of people's life, but it is also a significant skill to be taken into account in working teams.

Team members must be able to communicate, to tell each other what is going on, and the other part must be able to understand what his team mate is transmitting him. If communication fails, all the series of events after that failed communication are likely to fail too because of miscommunication.

Workers must not be shy among them, they have to be able to tell each other all the important toughs related to the development of the project in order to be able to improve it; if some worker does not say something because of his shyness and no one else notices it, the lack of social skills could end up in problems in the project and losses for the company.

A person with high social or interpersonal skills is able of listening and understanding what the others are saying, not only talking. He must be capable of choosing wisely the words the will use, in order to avoid misunderstandings and in case the interlocutor does not understand what has been said, think about why he didn't understand, and try to improve the communication.

Of course, for communication to be effective, the person must try to remain as calm as possible when trying to expose something, he must be able to speak quietly and clearly so it would be much easier to understand him and his message.

4.1.7. Negotiation Skills

Negotiation skills are highly related with communication skills, a person which has the ability of effectively communicate will have higher probabilities of being understood in a negotiation, and therefore of selling their idea to the people listening to him.

A negotiation occurs when two or more parties have a point of disagreement, and these two parts have to converse on how to reach an agreement which leaves both of the parties affected satisfied.

A good negotiator must have high communication and interpersonal skills which together, will help him to accomplish his objectives. He also needs to be a persuasive person, capable of convincing the other party to decide the most suitable outcome for him.

Summarizing, it is encourage that the worker is capable of negotiating, persuading, and thereby influencing the others, what is sought is a person which is capable of listening to the other part opinion and argument while at the same time he is influencing him into thinking and deciding what is more suitable for him.

People have to be careful on the way they handle their negotiations, it is not always about winning, but sometimes is better to find a fair settlement in which both parties are quite satisfied and then avoid the possible resentment of the team workers in case the resolution of the negotiation did not seem fair. The essence is finding the proper balance and focusing on a solution in which everybody wins.

4.2. Technical skills

In this section, technical skills more focused on software development but not exclusive from this field will be listed and described; these skills are no longer related to the personality of the worker and their personal abilities but they are critical to the development and success of a project.

4.2.1. Programming Skills

In every software project, programming skills are going to be needed, if something is going to be developed, several programmers will be necessary to develop it.
In order to develop programming skills the worker has to have an wide interesting in programming, and he does not only have to know a programming language, but several.

The ideal programmer should master a few programming language, know several ones, and have the capacity and the will to adapt to unknown or new programming languages that may come in their way. It is not only how much you know, but how much can you learn and how fast and good can you do it.

Nowadays, there are a lot of programming languages, and a lot more which are being created, but the most popular and useful ones that the workers must control and understand would be: Java, C, C++, C#, Python, Ruby...

Of course, web programming languages cannot be forgotten, as more and more applications are becoming web-based, to spare the user having to install the programs locally; so for this purpose, the main languages useful for a web developer must know are: PHP, Javascript, HTML, CSS, ASP and of course Database management knowledge is highly encouraged.

Finally, it is important to mention mobile applications, whose popularity has increased enormously over the last years and keeps growing rapidly. The major mobile operating systems to be considered right now are iOS and Android, so it is a great asset for the worker having wide knowledge in these operating systems and in designing and programming apps.

4.2.2. Analysis and Design Skills

In software development, analysis and design phases mostly consist on defined how the system is going to be, what it is going to need, how it is going to work and interact… and all of these have to be done before actually developing the system.

It is easy to understand that Analysis and Design skills are related to the ability of the user to propose solutions to a problem, and to design how this solution is going to be like in a future development process.

In order to be outstanding in this skill, the person must be organized, pay attention to details. He must be able to abstract the problem and extract from it all the different requirements which are going to be necessary for the development of the project.

Regarding design, highly related to creativity, the person has to guess how the system would look like, how the interaction between the user and the system are going to be and design this in an efficient and user friendly way.

Chapter 5 | Software Development Methodologies

In this chapter and the following subsections, an extensive definition of what a Software Development Methodology is will be provided, as well a list with useful information about the most commonly known methodology, emphasizing on the phases they have, advantages, disadvantages, and the kind of projects and teams for which they are more suitable.

A methodology is a systematic way of doing some kind of task: “A body of practices, procedures, and rules used by those who work in a discipline or engage in an inquiry; a set of working methods”.

From this definition, we can understand that a methodology is a set of guidelines and ordered steps to follow in order to accomplish a particular objective, regardless of the discipline. It is a set of practices to follow, not an algorithm.

If we apply this definition to the field of Software Engineering, we could say that a Software Development Methodology is a set of directives to follow which are used in order to structure, plan and control the whole process that involves the development of any kind of Software.

Nowadays, there is a huge variety of Software Development Methodologies, each of them with their different approaches on how Software Development Process should be handled, and which specific techniques (if any) are strongly encouraged to use in each phase. These methodologies can be more or less traditional, complete or partial and their different approaches can be very varied.

5.1. Traditional Methodologies

Traditional software development methodologies, or heavyweight methodologies, are based on a sequential series of steps and are those ones which impose a disciplined process, with a great emphasis on planning and strict and detailed documentation. In these methodologies the final Software Requirements need to be known and defined from the beginning, as changes will not be accepted afterwards.

The main characteristics of traditional methodologies are a predictive approach, comprehensive documentation and that they are process and tool oriented.

A predictive approach implies that, the development of the system is predictive and repeatable and everything is planned beforehand, from the project budget to the system's model.

When we talk about comprehensive documentation, it means that all the different processes have to be documented, and we are usually refer to requirements documentation (Big Design Upfront Process) as one of the most important pieces of documentation of the project.

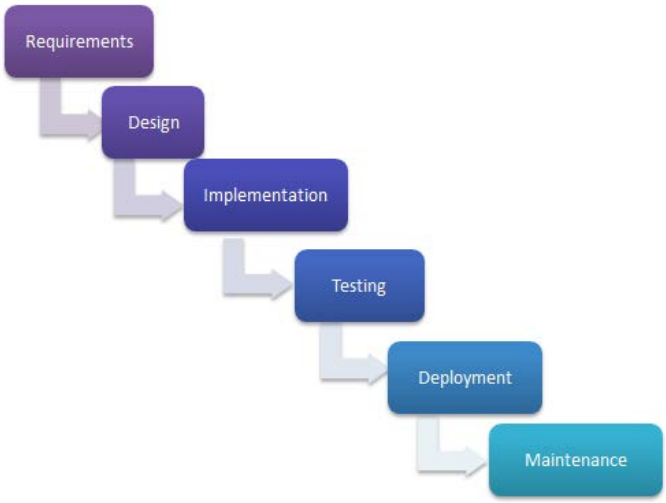
As traditional methodologies are process oriented, is the system the one which has to adapt to the methodology and not the opposite, as a consequence of this, all processes must be the same regardless of other factors. Of course, all these processes must be documented and tracked with the help of specific project management tools in order to record and update every process.

In the following subsections, three of the most popular heavyweight methodologies will be explained and detailed so as to understand better the different requirements, advantages and disadvantages of these methodologies.

5.1.1. Waterfall

Proposed by Winston Royce, the waterfall methodology is rigid, linear and sequential, and it is manly based on a structured progression between phases in which a new phase is not started until the previous one is finished and there is no return.

The different sequential phases of the waterfall model are the following:



1.Requirements
2.Design
3.Implementation
4.Testing
5.Deployment
6.Maintenance.

Figure 1 - Waterfall model phases

The output of the first phase is the input for the second phase and so on, and once a phase is completed there is no turning back.

This methodology is most appropriate for small projects when the requirements of the system are fixed, clear, extensively documented and there are no ambiguous requirements. For this to happen, the product definition has to be stable and stay the same along its development as well as the technology which is going to be used for the development of the project.

The main advantage of the Waterfall traditional model is that it is simple, and easy to understand. As each stage of the process is clearly and defined, strictly controlled and later changes are not allowed, it also allows departmentalization. The other great advantage that Waterfall is that absolutely everything done is documented, so there is always a record of all the steps followed during all the different phases of the project.

But, of course, this methodology has its disadvantages, and they are not few. The first drawback is that, as there is no working software until the very end of the project lifecycle, there is a lot of risk and uncertainty, as it could have been done all wrong from the beginning but it could not have been checked. In addition to this, it is not easy (if possible) to adapt to possible changes, so requirements cannot allow themselves to change either, which rarely happens in the real world.

As it can be seen, Waterfall is a very rigid methodology which should be used in very specific scenarios and preferably by meticulous people which are experts on what they do, in order to prevent and avoid changes in late stages of the product development.

5.1.2. Spiral

Defined by Barry Boehm, the Spiral model arises from the combination of some aspects from the Waterfall model (previously explained) and the idea of prototyping (iterative development), being highly focused on risk analysis.

The most attractive and useful feature of this methodology is the ability to identify and reduce the project risks in each iteration. This is done by developing incremental releases of the product with its corresponding incremental assessment and refinement done at each iteration.

Unlike other methodologies documentation is not all written upfront, it is created when needed and progressing along with the project's progress; if the project is incremental, so is the documentation.

Each time there is a new spiral (iteration), the prototype that has been released has to go through the following phases in order to move to the next one:

1. Determine objectives, alternatives and constraints
2. Identify and resolve the risks
3. Development and testing
4. Plan the next iteration

Figure 2- Spiral Stages
(http://commons.wikimedia.org/wiki/File:Software_Development_Spiral.svg)

This methodology is widely used due to the fact that it is more natural to develop small prototypes, evaluate them, correct them and then move on to the next step of the development process, reducing risks as much as possible and making easier to add changes in the middle of the development process thanks to its flexibility.

It is specially recommended for projects in which risk evaluation is an important item, and there is a limited budget which cannot be exceeded, so every possible setback should be detected and removed as soon as possible to avoid losses. This model is also suitable for projects in which the requirements have not been clearly defined, or they are too complex to be fully defined from the beginning of the development and as a consequence, changes are expected; the same happens with long-term projects, whose priorities and requirements are likely to change over time. As risks are evaluated constantly, it is recommended for medium and high risk projects, as everything will be under control.

In order to be able to achieve these objectives the human factor is considerably important in this methodology, the customer has to be highly involved along all the process to evaluate the various prototypes; and of course working team members should be highly skilled and experienced with emphasis on risk assessment techniques. Also, a team member with management skills and committed is necessary to be able to handle the complexity that this methodology implies.

Now, moving on to the advantages that this methodology offers, it is important to mention that the customer is able to see early versions of the final product thanks to prototyping, which makes them easier to see the evolution of the project and to decide to ask for changes in the earlier stages rather than at the end of the development. As the development is incremental instead of fully sequential, unexpected changes are more easily adapted and evaluated and requirements can be refined along the different iterations. The parts of the project which may imply more risks can be developed earlier so they are more exhaustively evaluated to minimize the risks on the most critical parts of the product.

On the other hand, all these advantages also entail a series of disadvantages or issues to take into account before opting for this methodology. The first of these drawbacks is that managing this kind of projects is difficult, and costly, as there are multiple iterations to manage and expertise skills are necessary. As all the processing and management are both complex, this methodology is not recommended for small projects as it would turn out into an expensive project. Finally, this development methodology is grounded on the idea of prototyping, but it could be the case that en of the development is not known, or that it may even never reach its ending and get stuck in intermediate iterations.

In short, Spiral way is a more flexible methodology for large and risky projects which allows the end user to obtain prototypes of the final product, knowing that risks are being managed and reduced and with the possibility of modifying requirements along the process.

*5.1.3. Rational Unified Process (RUP)**

The Rational Unified Process (RUP), developed by Rational Software and acquired by IBM, is an adaptable process framework, and it is a specific implementation of the Unified Process (UP) and allows process customization.

This methodology is somewhere in the middle between traditional and Agile methodologies; it could be considered traditional because of its weight, but because of its iterative approach could also be taken as an Agile methodology.

The aim of RUP is to increase team productivity by defining their roles and by providing common knowledge, language and tools to all the different team members, regardless their role on the team they have easy access to the same knowledge base.

The fundamentals of RUP are collected in the "six best practices", which are the following:

- Develop iteratively
- Manage requirements
- Employ a component-based Architecture
- Model software visually
- Continuously verify quality
- Control changes

RUP processes are divided in two dimensions, the first one related to time and the second one belonging to the fixed aspects of the project.

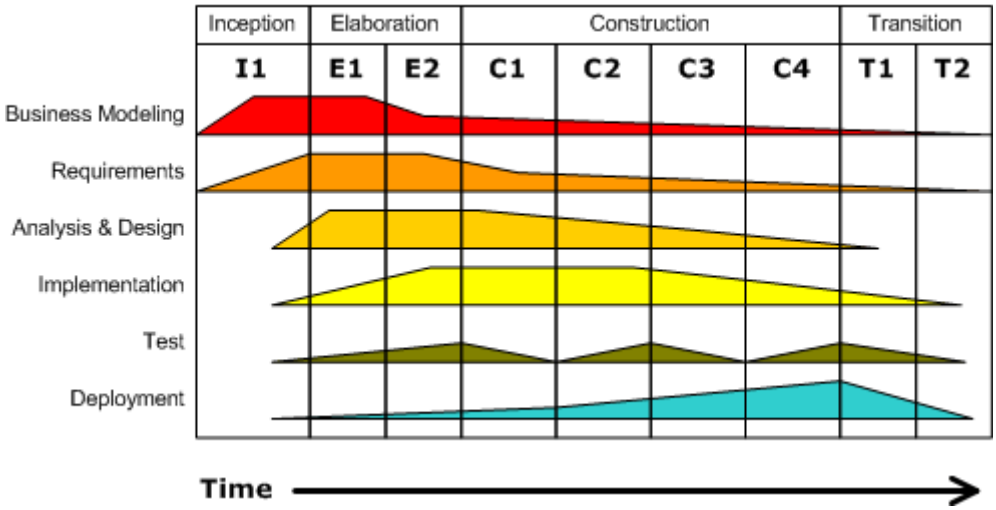


Figure 3 – RUP (http://commons.wikimedia.org/wiki/File:Development-iterative.gif)

RUP dynamic dimension is divided into four different phases which end at a specific milestone; these phases are inception (5%), elaboration (20%), construction (65%), and transition (10%).

During these four phases, RUP supports some or all of the disciplines displayed in the figure above, during the inception phase Business Modeling and Requirements are the main disciplines, and from elaboration to Transition the other disciplines such as Analysis, Design, Implementation, Testing and Deployment gain more importance.

As it can be seen in the image above, the testing discipline is always present, coming and going, as RUP is an iterative process, and the different iterations on the various phases have to be tested in order to reduce all the possible risks at early stages of the development.

Now we know a little about RUP, we can say that it is complete and very flexible, and could be able to adapt to any kind of project size, although it is most recommended for medium to large projects. Additionally, there is a lot of documentation available for team members making it easier for everyone to work in an homogeneous way. As risks are discovered during integration, it is easier to remove them, it is also easier to adapt to possible changes and the reuse of components, reducing the required development time.

One of the setbacks of this framework is the essential need of expert team members and even with the experts; many complications can arise because of the complexity of the process and the difficult organization. Another issue to take into account is the integration process, which may be hard for large projects if everything is not clear.

5.2. Agile methodologies

Once concluded with the introduction to traditional methodologies, we move on to a completely different field, Agile methodologies. As its own name says, these methodologies are distinguished by its Agile and lightweight approach, based mainly on iterative and incremental development.

Unlike traditional methodologies, Agile accepts the unpredictability factor, and that changes are usually in the middle of the development, this is way these methodologies are prepared to adapt to future changes in a way that heavyweight methodologies were not.

But, were did Agile come from? Who started to think about the need of a change from traditional methodologies? In 2001, the “Agile Manifesto” was written by a series of software developers in order to define and establish the basis of Agile approach.

Some of the main ideas about Agile development which are collected in this manifesto are that the interaction between people on the development team, as well as the communication with the customer and his involvement on the whole process, it is fundamental and a key aspect to determine the success or the failure of a project; it gives more importance to be able to present and provide working software to the

customer than to provide an extensive documentation work with no software and of course, the ability to respond to challenges in a quick and efficient way.

In order to fully understand the different Agile methodologies which will be provided in the next subsections, it is important to present the twelve principles in which Agile methodologies are based:

1. Satisfy the customer with continuous and early deliveries of quality software.
2. Deliver working software early (from a couple of weeks to a couple of months).
3. Welcome changing requirements.
4. Business people and developers must work together.
5. Build projects around motivated individuals.
6. The most efficient and effective way of communication are face-to-face conversations.
7. Working software is the primary way to measure of progress.
8. Promote sustainable development.
9. Continuous attention to technical excellence and good design.
10. Simplicity is essential.
11. The best Architecture, requirements and design emerge form self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjust its behavior accordingly.

These twelve principles are the main basis to look up for when using an Agile methodology on a software development process, besides the unique features of each different Agile methodology.

5.2.1. Lean Development (LD)

Adapted from the Toyota Production System, the main objective of Lean Development (LD) is to remove everything that it is not necessary for the project, to remove the waste and focus on the important and fundamental aspects of what the customer really wants.

Lean philosophy is based mainly on these seven principles:

1. Eliminate waste: Everything that does not add value should be eliminated.
2. Amplify learning: Short iterative cycles and provide feedback.
3. Decide as late as possible: Do not rush into (wrong) decisions.

4. Decide as fast as possible: Minimize the time to find a problem with its solution.
5. Empower the team: Allow the team to success.
6. Build quality in: Validate and re-validate and if it has no value, discard it.
7. See the whole: Pay attention to the interactions, not only the components.

This is a methodology which is more focused on the business and managerial aspects of the project development than the technical ones, especially those aspects related to Return on Investment (ROI).

As LD wants to obtain the right and exact requirements, a key and fundamental factor is the customer involvement along the development process because, at the end, what you want is the customer to be satisfied and by being involved he can understand the process and give feedback about the requirements and the business value along the whole process.

In addition to communication between the customer and the working team, it is also essential that the project managers know (and if not, they should be taught) how to communicate with the developers, in order to decide which step to take next, provide improvements or express doubts about the process. The customer should not only be in constant contact with the managers, but also with the developers, as they are the ones at a more technical level whose contributions are also important for the customers.

As Lean does not have a fixed structure of steps to follow as traditional methodologies, and is more focused on management than on technical issues it would be ideal to combine it with other Agile methodologies which could complement it, such as Extreme Programming, which will be explained later.

The advantages that Lean Development offers are clear, it is change-tolerant, reduces the waste to the minimum, keeping the requirements that add value to the product assuring the ROI, and where technical knowledge is not that important, so it could easily work with cross functional teams, recalling the importance of communication at all levels.

But again, no methodology is flawless, as good as it is that Lean does not forces an specific technical development process it is also its weak point, because nothing technical is defined and should be the team the one choosing the process to follow according the Lean principles. Additionally, the success of the project depends too much on the quality of the communication between all the parts, and their cohesion, so

this methodology should be limited to small groups in which such strong communication is possible.

5.2.2. Feature Driven Development (FDD)

Feature Driven Development (FDD) is an incremental and iterative software development methodology conceived by Jeff de Luca whose practices are driven from the perspective of the features of the system and with short-iteration deliveries. Absolutely everything is planned and done in a per-feature basis.

As it can be guessed, the basic unit of work in this methodology is the feature, and it is expressed in the following form:

```
<action><result>[of|to|for|from]<object>.
```

The foundational step for FDD is creating a model of the domain, this means that it is necessary to collect knowledge from all domain experts and then integrate it into a unified model which will represent the initial problem domain. Once evaluated the model, it is necessary to plan the needed resources and finally a small set of features and subject areas are identified and assigned to a team to work on them and when this set is done, then the next set of features must be developed. In the following figure, a summary of the processes to be followed is provided:

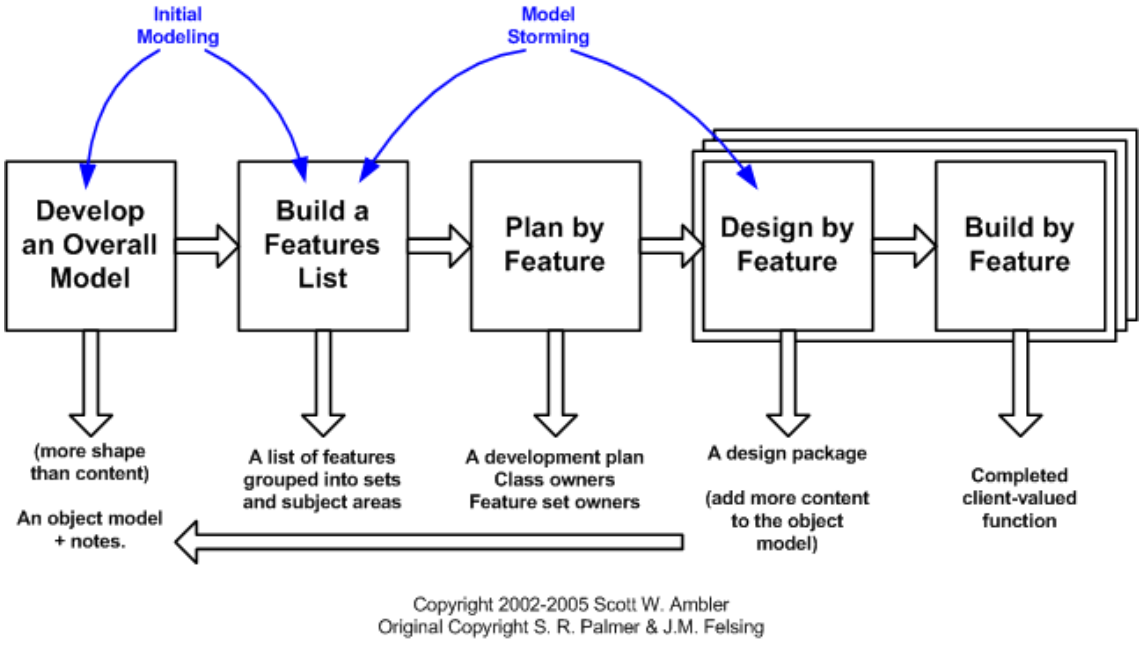


Figure 4 - Feature Driven Development (http://www.agilemodeling.com/essays/fdd.htm)

Regarding the roles distribution in FDD, there are six main roles to highlight, which are the Project Manager, Chief Architect, Development Manager, Chief Programmer, Class Owner and the Domain Expert and one person can have more than one role in the project, but classes (or pieces of code) are assigned to one individual, in a way that Class Owners have to work together in case some feature requires the modification of several classes.

In order to track the current status of all the several features of a project, FDD has various milestones to capture where these features are and show this progress to the customers:

1. Domain Walkthrough
2. Design
3. Design Inspection
4. Code
5. Code Inspection
6. Promote to Build

A unique feature of FDD is the Feature Set Progress Report which, with the aid of color codes shows the state of each Feature Set regarding to its completion and including information such as the Chief Programmer in charge of each Feature Set.

Because of its way of functioning with Code Owners, FDD supports multiple teams working in parallel, which reduces considerably the development time required. Additionally, “feature” is a concept very easy to understand and to identify, making their localization and development easier for all kind of team members and as, everything is driven by feature, and it is a flexible methodology in a way that it is suitable for volatile requirements, which may change easily. Finally, the other great benefit of using FDD is that it scales easily, for both large and small projects and team sizes, as well as for new projects or existing ones, but this scaling is usually limited by the number of Chief Programmers.

On the other hand, iterations may not be always be well defined and this may lead to confusion; also, as everything is feature-driven, some other aspects which don’t are not directly related with features may be left aside, such as design or maintenance. Related to teamwork, as there is no shared code, communication between Code Owners may be

less fluid than with other Agile methodologies in which almost everything is shared among almost everyone, fomenting individualism and forgetting about teamwork.

5.2.3. Dynamic Systems Development Model (DSDM)

Originally developed as an extension of Rapid Application Development (RAD), Dynamic Systems Development Model (DSDM) is one of the most heavyweight Agile methodologies. It is focused on the business perspective, and its objective is to provide business solutions, in a fast and efficient way without forgetting about budget limitations.

An important factor about DSDM is that every part of the project is divided and classified according to its priority, with a must, a should, a could or a would, depending on its characteristics. In DSDM time and budget are fixed values, while requirements are the variable ones and are allowed to change.

This Agile methodology is based on constant communication with the customer, whose involvement, collaboration and cooperation during the process are required, and this is encouraged by frequent deliveries of iterative increments of the project. The deliverables have to meet the business requirements and being able to accept changes, taking into account that all these changes must be reversible and it is possible to go back to a previous state.

Another fundamental aspect that must be present is testing, the way to test is not strictly defined, but it is a must that tested are involved in every aspect of the process to assure quality and control, so each development team should have a tester with them.

DSDM ideal team size is said to be teams of six members, scaling up to six working teams (of six members each). Each team should count with a tester in order not to compromise the quality of the final product and business knowledge is highly required for all team members.

Regarding the phases and steps that have to be followed on DSDM:

- Pre-Project.
- Project Lifecycle. Inside it here are several stages where the last three repeat a certain number of iterations:
    - Feasibility
    - Business

- Functional Model
- Design and Build
- Implementation

- Post-Project.

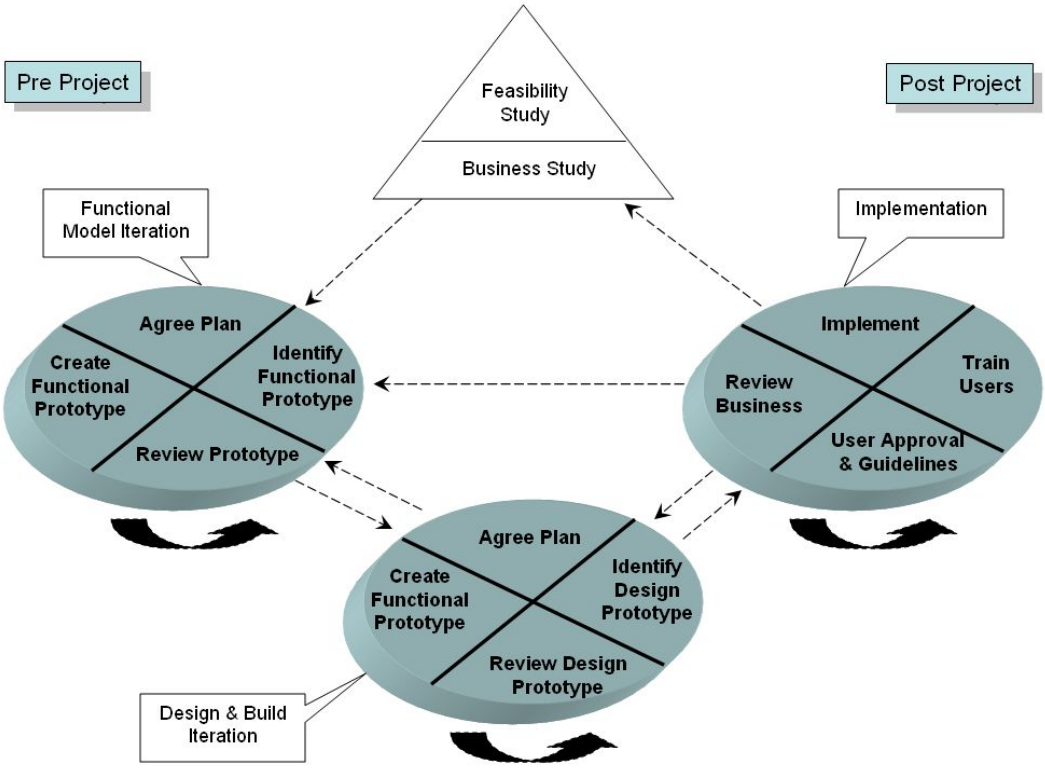


Figure 5 - DSDM (http://assets.devx.com/articlefigs/17425.jpg)

Once understood the requirements and the functioning of DSDM, we are now able of discerning between its pros and cons. Among the benefits of using this methodology, is that the emphasis on constant testing probably pays back in a product with high quality products, while keeping the business value at the top of the priorities list. Another advantage that DSDM has with respect other methodologies, is that the customer is conscious that not all the requirements may make it to the final product, but that the product will be ready on time and with the agreed initial budget.

As an Agile methodology, it could be a little heavy, as there are a lot of steps to follow, classifications to make, and the documentation load is heavy too. As it requires constant communication among team members, and the development teams should have no more than six members to keep this communication fluid, this methodology is not recommended for distributed teams, as this would slow down the communications and

the working process. It is neither recommended for high criticality systems, as they probably would request some fixed requirements from the beginning and we know now that for DSDM the variable are the requirements, not the time nor the budget.

5.2.4. Scrum

Scrum, maybe one of the most known Agile methodologies, the sentence which summarizes its goal is: “Prioritized Business Value”, its objective is to deliver software that after each different iteration, offers the highest business value. Scrum is a framework that is lightweight and easy to understand, but quite difficult to master. As it does not specify technical practices to follow, Scrum is a methodology which is easy to integrate with other complementary methodologies more focused on engineering practices.

Scrum is an Agile methodology which is also based on incremental deliveries on several iterations, each iteration has a duration of 30 days or less and it is called a Sprint. The other key factor of Scrum is the team organization, they have to organize themselves based on the Sprint deliveries and also, team members will have to meet every day for the scrum meetings.

These meetings consist of 15 minute reunions in which team members gather together in order to answer individually three different questions so as to update the other team members on the development status, these questions are:

- What was done since the last meeting?
- What is my plan between this meeting and the next one?
- Is there any roadblock?

With these meetings, individuals do not report the status of their assignments in term of completion, but on what they have done, if the current task has been completed or not. In case there were any roadblocks, the team should decide which the best way to handle them is.

In case there was more than one working team for the same projects, the Scrum of the Scrums would take place, in which one member of each development team would gather with the other representatives from the other groups and have another Scrum meeting together.

Scrum teams are formed by the Product Owner, the Development team and the Scrum Master and, as we said before these teams have to be able to organize themselves and they are also cross-functional teams.

The Product Owner is the voice of the customer; he is a sole person responsible for the product's value, and for maximizing it. The Product Owner is in charge of managing the product's Backlog while he can also be part of the Development team.

The Development team is responsible for developing and delivering quality potential product on each different increment of the development process. Development teams work as a whole individual, they may have different skills but at the end they are all developers. It is important to find a balance on the number of members inside the Development team, the ideal cipher would be from 3 to 9 members, small enough to ease the communication and coordination but big enough to be able to perform a significant amount of work.

Finally, the Scrum Master is at the same time the leader and the servant of the Scrum team; it is his responsibility to make sure that Scrum principles are fully understood and that they are being followed each step of the process and as he is in constant communication with the Development team he is also in charge of the interaction with the customer.

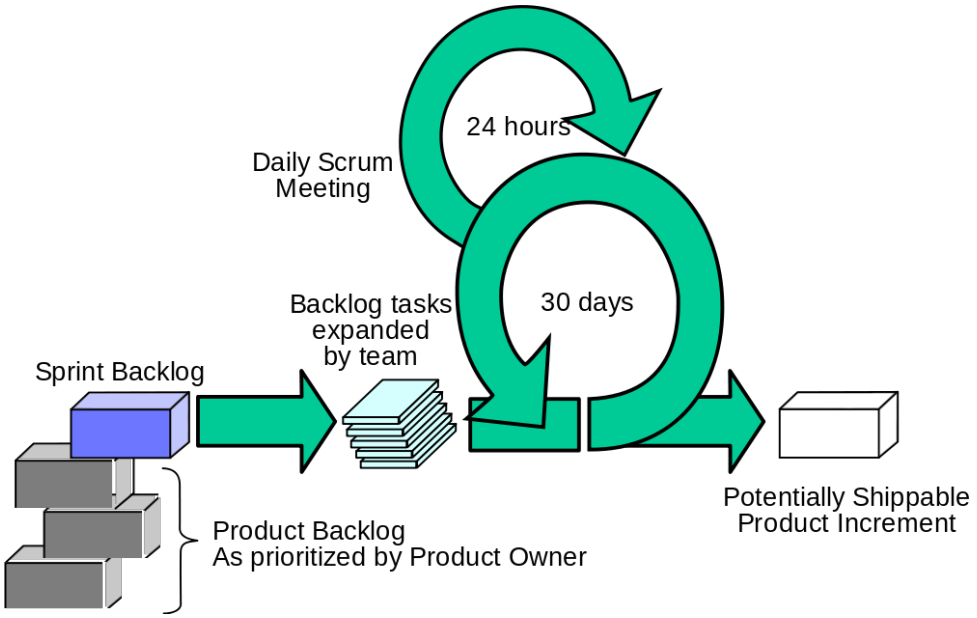


Figure 6 - Scrum (http://commons.wikimedia.org/wiki/File:ScrumSchwaberBeedle.svg)

In order to provide transparency and a chance for future adjustments, Scrum methodology counts with two artifacts, the Product Backlog and the Sprint Backlog. The Product Backlog is a list containing everything that would be required for the product development ordered by priority, and the only member allowed to modify this Backlog is the Product Owner; this list evolves in a dynamic way along with the project as new needs are discovered. The Sprint Backlog contains a set of requirements from the Product Backlog which are selected for a particular Sprint and detailed information on how to achieve these goals; as with the Product Backlog, as requirements are fulfilled, changed or considered not necessary the Backlog is also modified.

After this extensive introduction to Scrum methodology, we are able to determine which are the kind of projects which are more appropriate for using Scrum instead of other methodology. Scrum is extremely useful for new projects which have not been done before and whose estimations are likely to change during the development of the product, that is to say, for highly volatile requirements. As it was explained before, Scrum is ideal for small teams, which could be distributed as long as they hold the required meetings and keep on constant communication and dedication to the project. Because this methodology is only focused on project management aspects, it is fundamental to find a methodology to use it with and compensate the technical lacks of Scrum.

5.2.5. Extreme Programming (XP)

Along with Scrum, Extreme Programming (XP from now on) is one of the more well-known Agile methodologies. Created by Kent Beck, it is easy to imagine that this methodology is focused on programming and on the programmers, as it emphasizes technical and defined practices.

As other Agile methodologies, XP lifecycle is based on frequent deliveries of quality working software, but unlike most of Agile methodologies, in which the techniques are no so important or not so strictly defined, in the case of XP the technical practice is taken to the “extreme”, hence its name.

The four core values which sustain the skeleton of Extreme Programming are communication, simplicity, feedback and courage. From these four core values, we can derive 13 fundamental practices in XP, which I will separate in 5 different core groups to make their understanding easier.

- **Whole Team**: The team has to work as a unique entity, every individual is a part of the Whole team, and being generalized rather than specialized is strongly encouraged, so it is important to try to learn as much as possible from everything.

- **Planning Game, Small Releases and Customer Tests**: The work should be planned in an incremental way, delivering releases as quick as possible and with constant communication with the customer for early feedback on the deliveries, better understanding of the requirements and faster work.

- **Simple Design, Pair Programming:** The design to be used must be the simplest one that works for what it is required, avoid doing this that may never be needed, in order to do this, team size recommended is two or three, in order to share opinions quickly and reach faster to conclusions.

- **Testing, Refactoring, Continuous Integration:** everything must be tested, and it is encouraged if these tests could be automatic. The design must not be all done upfront, XP encourage designing as the project progresses and making improvements when it is needed; for that reason the code is being rebuilt constantly and has to be checked properly on each integration phase, hence, the importance of continuous testing.

- **Collective Code Ownership, Coding Standard**: In XP, the code has no owner, everyone can touch everything if he is in the development team and for this reason, and in order to avoid communication problems, a set of coding standards must be previously defined by the team members in order to develop the code in a uniform and standard way.

- **Metaphor, Sustainable Pace**: The XP team is encouraged to have a metaphor to describe the idea of the system and the work done by the team members should be reasonable, they have to work hard, but only up to the point it is productive to do so.

All these 13 practices support and relate to each other as the following figure represent, so we can which practice depends more on other practices and the possible consequences of changing, removing or adding a practice.

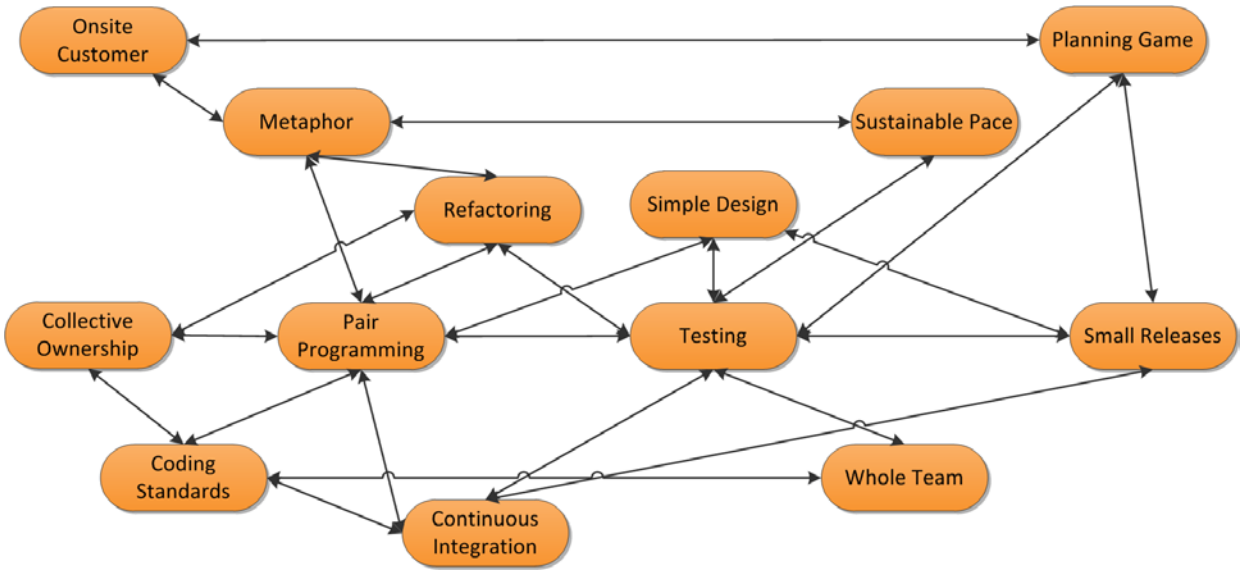


Figure 7 - XP 13 practices

Regarding the game planning techniques (mentioned in the 13 practices) used in Extreme Programming, there are two modalities used: the release and the iteration. Games are meetings that take place every each iteration, which usually happens to be a week.

- Release planning: in this plan the developers and the customer are involved, and the requirements which are going to be delivered in next deliveries are defined, as well as their delivery deadlines. This planning is composed of three stages:

- Iteration planning: in this plan only developers are involved and it is when the plans and tasks of the different developers are scheduled and planned.

Each of these planning games are also composed by three different stages which are Exploration, Commitment and Steering.

5.3. Conclusions on methodologies

After reviewing and analyzing some of the most important traditional and agile methodologies, some relationship between the methodologies and human capabilities were extracted.

The following tables provide a correspondence between the methodologies studied in this project and the most important human capabilities described in the previous chapter.

The matrixes establish the most important capabilities for each of the methodologies analyzed; depending on the methodology studied one, two or three capabilities have been selected as the more critical and the conclusions on these relationships between methodologies have been extracted from the information read and collected for Chapter 5 on all the methodologies.

This information will be used in following sections and it will help the genetic algorithm at the time of configuring the best working team when using a particular methodology.

Capability/Methodology	Waterfall	Spiral	RUP	Lean	FDD	DSDM	Scrum	XP
Team Worker				X		X	X	X
Leadership								
Commitment		X	X		X			
Planning	X							
Creativity								
Social Skills						X	X	
Negotiation Skills						X		
Programming Skills			X					X
Analysis and Design Skills		X	X		X			

Table 1 - Methodology Classification Chart 1

Chapter 6 | Genetic Algorithms

Algorithms are procedures performed on several organized steps which lead the way to solve a specific problem; they are used for data processing, calculations and automated reasoning, among others.

The concept of Genetic Algorithms (Gas from now on) emerges from the hand of John Henry Holland in the 1970s; they are adaptive methods which are used solve search and optimization process, as the one presented in this project; but what makes this algorithms special is that they are inspired by and based on the genetic processes of living organisms and that they are based on the theory of natural selection and evolution.

This inspiration for GAs comes from the nature of individuals of fighting for the resources needed to live; the individuals with more success are the more likely to survive, get a partner and have the greatest number of offspring; the weakest individuals or the less gifted ones have smaller chances of having offspring and so the children of the strongest ones will be the ones with more chances of spreading.

GAsare an analogy to natural behavior, and each solution is represented by a population. Inside a population, each different individual has a score and, as we can imagine, the greater the individual's score, the greater are the chances for this individual to be selected to reproduce and produce offspring with another individual with high score so as to generate new populations with a higher proportion of the best features in regarding the previous population. If the GA is properly designed, there will be a point where the resulting population will converge to an optimal solution for the proposed problem.

The strength of GAs is mostly due to the fact that they are a robust technique, they can successfully solve problems in almost every type of situations, but the scenarios in which they shine the most are those ones in which the problems in need of a solution do not have yet a specialized technique to help to find the solution.

6.1. The Canonical Genetic Algorithm

The Canonical GA represents the generic functionality of a GA, and it is the basis for any other variations of these algorithms. The following flowchart contains all the steps followed by the algorithm which will be explained afterwards.

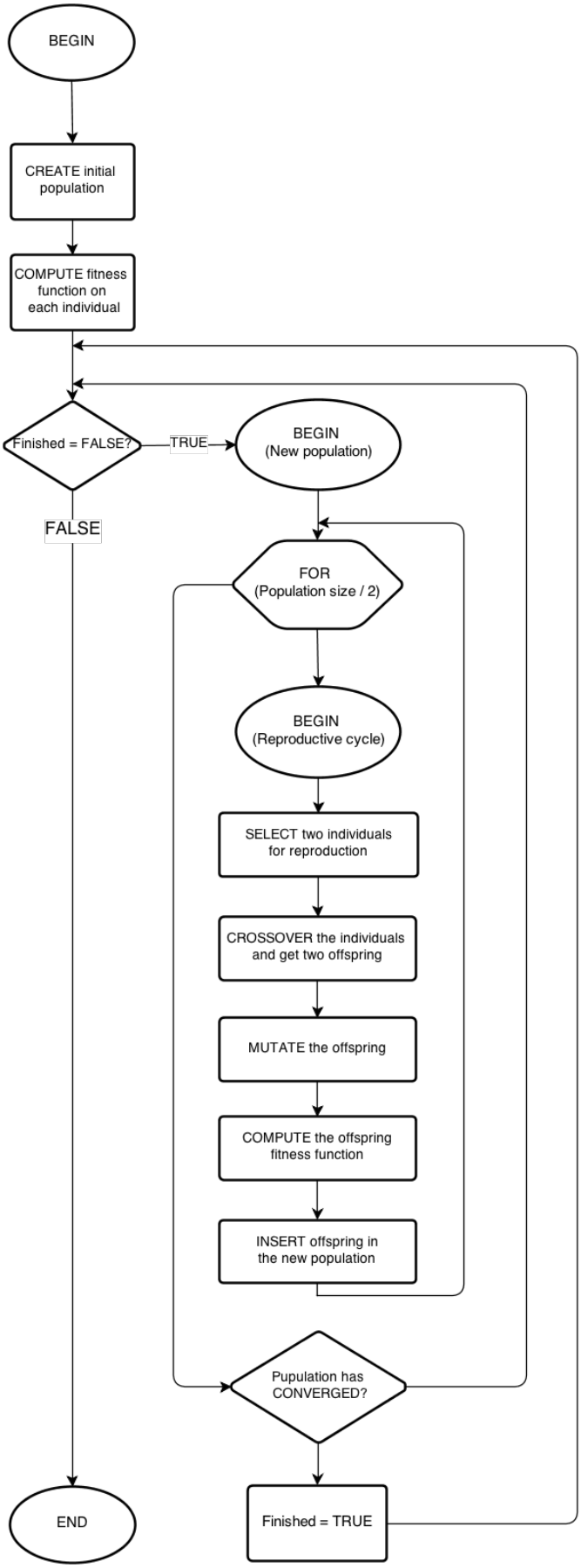

BEGIN
CREATE initial population
COMPUTE fitness function on each individual
Finished = FALSE?
TRUE
BEGIN (New population)
FALSE
FOR (Population size / 2)
BEGIN (Reproductive cycle)
SELECT two individuals for reproduction
CROSSOVER the individuals and get two offspring
MUTATE the offspring
COMPUTE the offspring fitness function
INSERT offspring in the new population
Pupulation has CONVERGED?
Finished = TRUE
END

Figure 8 - Canonical GA

As we can see in the previous diagram, the procedure of this algorithm is quite simple, an initial population is generated, and a fitness function is calculated on every individual to obtain the fittest ones, once done that, the fittest "parents" are selected for reproduction, and they are crossed with a certain probability; after the crossover, two offspring are generated and they are mutated with a mutation operator. After that, the fitness function is calculated for each of the new individuals and they will be part of the next generation of the population.

In the following section, the principal steps to be followed with GAs will be explained in detail in order to fully understand all the intricacies of GAs way of functioning.

6.2. Genetics Algorithm Methodology

In GAs, biological terminology is used in order to identify all the elements of the problem; the different individuals represent the possible solutions, those individuals can be represented as a set of genes, or parameters, which all together form a group of chromosomes, or values.

The set of parameters or genes that put together represent a specific chromosome, are called a phenotype, and it contains the necessary information in order to create an organism; that would be a genotype. For example, in biology, the genetic information contained in an individual's DNA would be the genotype while the expression of that DNA, in other words, the individual, would be the phenotype.

Generally, the initial population for the problem is randomly generated in order to emulate the real word, and the size of that initial population will depend on the amount of data to process or the complexity of the proposed problem.

6.2.1. Fitness Function and Selection

The fitness function has to be specifically designed for each particular problem; given an specific chromosome, the fitness function will be in charge of assigning a numerical value which reflects the level of fitness or adaptation of that individual for that particular problem, and that will be decisive at the time of deciding if that individual goes into the reproductive phase or not.

On the reproductive phase, two of the fittest individuals would be selected for the cross-over process and produce offspring, which, once mutated, will belong to the next generation of the population.

The way the two parents are selected is by a random procedure which will favor the fittest individuals, as each of them would have a probability of being chosen for reproduction according to the result of the fitness function on these individuals so, the probability of being chosen for an individual with a high fitness value would be greater than for one individual with low results on the fitness function. According to this, the fittest individuals may be chosen more than once for reproduction whereas the least fit may never be selected for the reproductive process.

Once the parents have been selected, their chromosomes will be combined by using the crossover and mutation operators.

6.2.2. Crossover

Crossover is a genetic operator used in GAs in order to modify and combine the chromosomes of the both parents selected for reproduction into the new chromosome of the offspring.

There are many possible ways to perform this crossover, the basic one would be the one-point crossover, which consists on selecting one point of the chromosome and, all the genetic information from that point is swapped between the two individuals generating two offspring with the genetic distribution based in that point selection for the crossover.

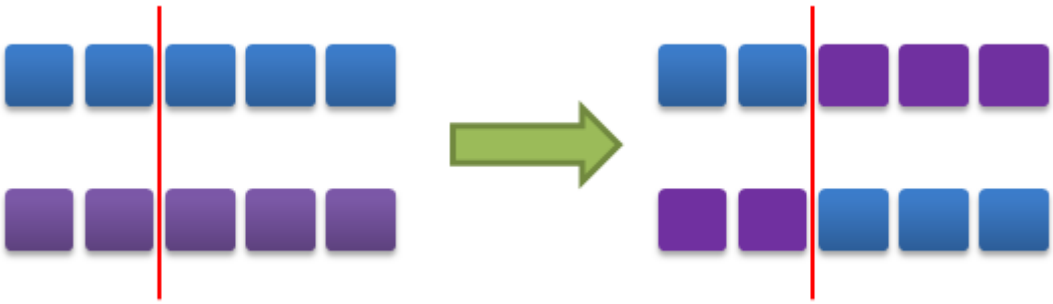


Figure 9 - One-point Crossover Example

Other possible way of performing the crossover is the two-point crossover, similar to the previous one, but this time everything contained between these two points is swapped between the parents obtaining two children with the following structure:

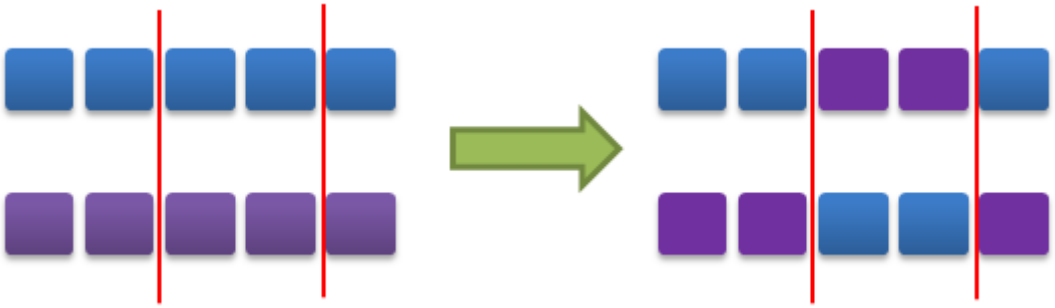


Figure 10 - Two-point Crossover Example

Now, another interesting technique for crossover is called the Uniform crossover and its objective is that both parents are able of contributing at a gene level with a more uniform distribution than with whole segments of genetic code as in the two previous cases.

This operator decides in which will be the mixing ratio, that is to say, the percentage in which each parent will contribute to the chromosome of the new offspring. If that mixing ratio is 0.5, approximately half of the genes of the children will be from the first parent and the other half from the second one; and remember, we are talking about individual genes, not about full segments.

Additionally, we also can use arithmetic crossover operators or heuristic ones. Arithmetic crossover consists on applying some arithmetic operation and performing a linear combination of both parents. In a simple example, if the operation chosen would be to simply add the binary values of each gene of each parent the result would be this:

Figure 11 - Arithmetic Crossover Example

Regarding heuristic crossover operator, this one is interesting because the mixing is not done randomly; this technique takes into account the fitness of the chromosomes of the different parents and takes into account these formulas in order to generate the two possible offspring:

```
Offspring 1 = Best Parent + r * (Best Parent-Worst Parent)
Offspring 2 = Best Parent
```

- r= random number between 0 and 1

The problem with this operator is that it can generate not feasible first offspring if the value chosen for "r" makes the value of the genes go out of their bounds and in order to solve his problem is setting a parameter which will indicate the number of tries to obtain a feasible result for the first offspring and when the attempts are over, the Worst Parent would be selected as the first offspring.

The problem with this technique is that it could be the case that there is no crossover process at all and the offspring are exactly like one of the parents, depending exclusively on the mutation process.

6.2.3. Mutation

The other genetic operator used in GAs is the mutation operator; its main purpose is to maintain genetic diversity from the chromosomes of one generation of population to the next one by altering one or more gene values of the offspring chromosome and it is usually done after applying the crossover operator on the offspring.

Mutation takes place according to a probability which is defined by the user, usually low because if it is set too high it could all turn into a randomization of the genes with no criteria; it alters the value of the genes from its initial state in order to arrive to a better and more optimal solution that the one already proposed.

The other main objective of the mutation operator is to prevent the population from becoming stagnant on local peeks and prevent the search from stopping when local optimal points are reached.

The simplest way of applying this operator when using binary encoding is by flipping a bit, it simply consists on selecting a random gene, and changing its value from 0 to 1 or from 1 to 0, depending on the given value.

Figure 12 - Mutation Example

For integer values, there is more variety regarding mutations possibilities, for example, the mutation method which is going to be used in the Genetic Algorithm for this project is the Uniform mutation, which replaces the selected gene with a random gene which is between some lower and upper bounds previously defined by the user.

6.2.4. Termination

As we have seen in the diagram of the Canonical Genetic Algorithm, the reproductive process is repeated several times until the termination condition is activated; the way to determine this termination condition is chosen by the user and could be for example, until a finite number of new populations have been generated, until a maximum of time or resources have been consumed and no optimal solution has been found yet, or the algorithm could stop when there is a solution which meets the minimum criteria of optimality and no more iterations are needed.

PART III:

ANALYSIS

Chapter 7 | Conceptual Model

Once finished with the State of the Art, we have now a global and more extensive knowledge of all the diverse topics which are important for the development and understanding of the current project and that will be present along the following sections.

A Conceptual Model could be described as a diagram which represents the relationships between the main entities or classes that compose this system, and all these classes are going to be explained and described along this chapter with the visual help of UML (Unified Modeling Language) diagrams.

In this particular project, we are talking about projects, methodologies, teams and workers and the relationship among these different entities may not always be clear. The formalization of a conceptual model for this system helps to put together these classes and to determine which are the relationships among entities, as well as which are the main components and attributes of each of the classes.

The proposed Conceptual Model defined with a UML diagram is one of many that could actually work for these different classes, and more interpretations of the same problem are probably correct as well. In the following sections and subsections these Model will be detailed and explained.

7.1. Analysis of the Conceptual Model Input Data

As said before, the data we need to analyze, and process to be the input data for this system, is the one related with the projects, the methodology employed, the team working on the project and the set of individual which form the team.

There were many possible way to correlate these classes together in a way that made sense as, at the end, each of them depends on the others at some level although there are some hierarchies which may look less ambiguous than the others. In the following figure we can see the class distribution as well as the relationships among them.

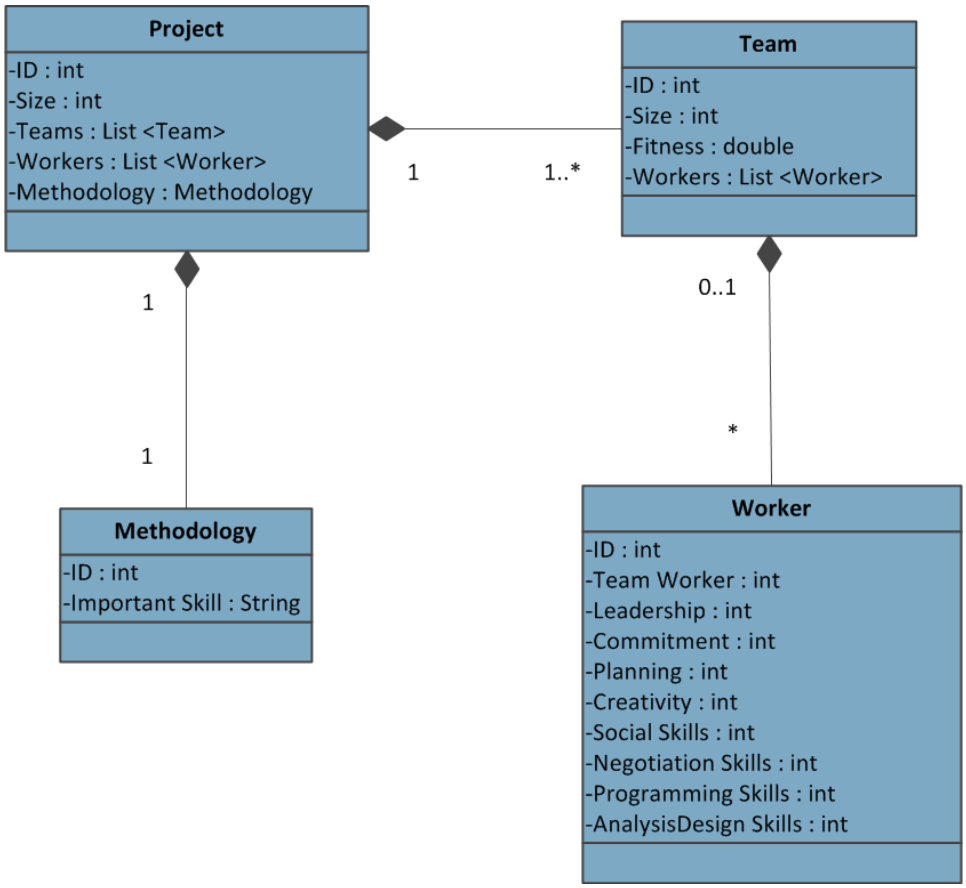


Figure 13 - Conceptual Model

In this figure, it can be seen that the central element of the Conceptual Model might be the project, but at the end all of the four elements in it are of vital importance for the proposed system.

In the figure we can see the attributes of each class, which will be explained in detail later; it can be observed that a project class has its own attributes, but at the same time is composed by the Methodology class and the team class, so a project is composed by a Methodology and an team (besides its own attributes).

Additionally, we can see other composition in this class diagram, as a team is composed by several workers; the team size can depend on the optimal combination of team members and methodology for the proposed project or can be defined previously by the user to be fixed.

Following, each of the four classes will be explained in further detail as well as the attributes which define them and the criteria used to extract and process the entry data for the system.

7.1.1. Methodology

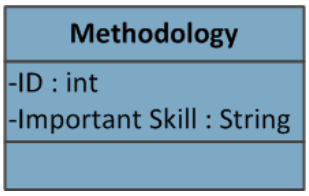


Figure 14 - Methodology

This first class, as its name indicates, represents the methodology which is going to be used for a particular project, it has a series of attributes which help categorizing the methodology and find the most suitable one for a particular project jointly with the team in charge of the development of such project. In the following table each of the attributes which are part of the class will be described.

Attribute	Description
ID	This attribute is used in order to identify the methodology by a unique ID. The different available methodologies are: Waterfall, Spiral, RUP, Lean, FDD, DSDM, Scrum and XP.
Important Skill	This attribute represents the most important skill of a worker for a particular methodology. The skills to be considered by the methodology are the ones contained in the worker class.

Table 2 - Methodology attributes

7.1.2. Worker

This second class, Worker, is the one in which every single attribute about a single worker is defined. All the information collected for each worker from the 360 reviews will be given this format, taking into account only these attributes for this particular project, but the scope of users' attributes could be expanded easily in future projects.

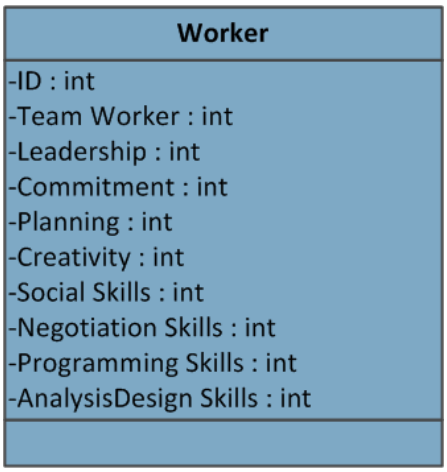


Figure 15 - Worker

Attribute	Description
ID	This attribute is used in order to identify the worker by a unique numeric ID, starting in 1 for the first worker.
Team Worker	This attribute represents the ability of the worker to work in teams, collaborate with other team workers and contribute as part of the group to the whole project, not only individually.
Leadership	This attributes punctuates the leadership skills of a worker, this means and the ability to influence and motivate other people in order to achieve further goals and to achieve the proposed objectives.
Commitment	This attribute is related to the dedication of the worker to the project, the level of commitment with the work and the discipline employed at the time of developing the project, this means, being there at all the development stages, and collaborating throughout the process.
Planning	This attribute reflects the organization level of the worker, if he is capable of doing accurate and realistic estimates, and follow them as far as possible without needing to rearrange deadlines.
Creativity	This attribute relates to the ability of coming up with new ideas or innovative concepts which can be useful in the development of a new project at the early stages.
Social Skills	This attributes makes reference to the ability of the worker of interacting with the others in every aspect of his life . The kindness to the others, communication skills and a good relationship with the co-workers are highly encouraged
Negotiation	This attribute describes the capacity of a worker, to be heard, to be

Skills	able to convince someone or a group of people that his solution is better than the others, or as least make the other listen to the arguments he has to offer for a particular project.
Programming Skills	This attribute defines the level of technical abilities of the individual regarding programming languages knowledge, fluency and experience. This is extremely important for projects which require a high level of expertise from the workers due to their complexity.
Analysis and Design Skills	The last attribute of this class determines the ability of defining how is the system going to be before it is done, determining the needs, requirements and the interactions which are going to be present prior the development and the ability of changing and adapting these requirements and designs as the project evolves with them.

Table 3 - Worker attributes

7.1.3. Team

This third class is a relatively small class; the Team class is included in order to put together all the different workers which will be part of a particular working team which will be competing with the other theams to be the one chosen for the current project.

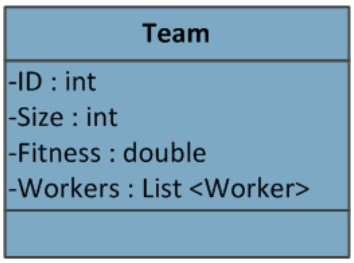


Figure 16 – Team

Attribute	Description
ID	This attribute is used in order to identify the team by a unique numeric ID. This ID is composed by an ordered list of the workers belonging to that group so it can be easily identified and will also be useful at the time of avoiding repeating teams.
Size	This attribute, as its name says, will define the size of the working team by representing the number of workers which will be part of the development team. This attribute should be as consistent as possible with the recommended team size attribute from the Methodology class in order to make the optimal combination of team and methodology for the proposed project. The team size would have a minimum value of two workers with no upper bounds defined.
Fitness	This attribute is used in order to save and determine the Fitness of a particular group. Depending on the workers belonging to the team and their own capabilities, the fitness of the team containing them will be different.
Workers	The last attribute of the team class is a list containing all the different people which will belong to the team, including all their different attributes. The list of team members should have a length consistent with the size of the team. A worker may or may be not part of the team depending on their particular capabilities in relationship with the methodology and the project.

Table 4 - Team attributes

7.1.4. Project

Lastly, there is the class which contains the information about all the described classes above and puts all the different information which can concern a Software Development project all together, the Project class.

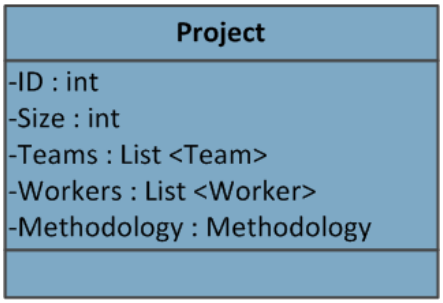


Figure 17 - Project

Attribute	Description
ID	This attribute contains the identifier of a particular project which will be numeric and unique.
Size	This attribute refers to the size of the proposed project and will be strongly affecting the team size (in accordance with the methodology). The different values proposed for this attributes are “Small”, “Medium” and “Large”.
Team	This attribute contains the teams available for a particular project
Workers	This attribute defines the list of all the available workers for a particular project
Methodology	This attribute specifies the methodology to be used in the proposed project.

Table 5 - Project attributes

Chapter 8 | Requirements definition

A requirement is something that is needed; the requirements definition stage consists on defining and described what are going to be necessary, how these needs are going to be fulfilled and the constraints surrounding them.

In this chapter, the different types of requirements which need to be described and evaluated will be provided in order to get a clear definition of the final functionalities of the system and its needs as well as the inner relationships among these requirements.

8.1. User Requirements

A User Requirement is a "condition or capability needed by a user to solve a problem or achieve an objective", this means, User Requirements are the different needs that the user will have at the time of interacting with the system in order to achieve his final objective, in this case optimizing working teams.

In the following subsections the main two User Requirements categories, capability and constraints requirements, will be explained and the different requirements will be defined and classified.

The fields which are going to be defined for every User Requirement are the following:

- **Identifier**: Unique ID of the requirement with the format "UR-XX", where XX determines the number of requirement.
- **Name**: Descriptive name of the requirement
- **Type**: Type of User Requirement (Capability or Constraint)
- **Priority**: Priority of the requirement being fulfilled (High, Medium or Low)
- **Optionality**: This field defines if the requirement is optional or not.
- **Description**: Brief description of the requirement

8.1.1. Capability Requirements

Capability requirements describe what the user wants to do with the system; the process that the user must carry out to fulfill the user needs and from these requirements, functional Software Requirements will be derived.

Identifier	UR-01					
Name	Register					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to register into the system.					

Table 6 - UR-01: Register

Identifier	UR-02					
Name	Log In					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to log into the system					

Table 7 - UR-02 Log In

Identifier	UR-03					
Name	Log Out					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to log out of the system					

Table 8 - UR-03 Log Out

Identifier	UR-04					
Name	Add a project					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to add a new project to the program					

Table 9 - UR-04 Add a project

Identifier	UR-05					
Name	Define project's attributes					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will have to define the different attributes for the project which are fixed (name, size)					

Table 10 - UR-05: Define project's attributes

Identifier	UR-06					
Name	Select Methodology					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory			Optional		X
Description	The user will have the option of selecting the methodology that is going to be used in the project					

Table 11- UR-06: Select Methodology

Identifier	UR-07					
Name	Highlight an Attribute					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		X
Description	The user will have the option of selecting a particular ability essential for the project that the team workers must have					

Table 12 - UR-07: Highlight an Attribute

Identifier	UR-08					
Name	Save project's information					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to save the current project's information for future uses					

Table 13 - UR-08: Save project's information

Identifier	UR-09					
Name	Load project's information					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to load the information of a project previously saved					

Table 14 - UR-09 Load project's information

Identifier	UR-10					
Name	Delete project					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to delete a whole project					

Table 15 - UR-10: Delete project

Identifier	UR-11					
Name	Add Worker					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to manually add potential workers for the current project					

Table 16 - UR-11: Add Worker

Identifier	UR-12					
Name	Define Worker's attributes					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to define the different attributes and competences of the current worker					

Table 17 - UR-12: Define Worker's attributes

Identifier	UR-13					
Name	Save Worker					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to save the information of the current worker					

Table 18 - UR-13: Save Worker

Identifier	UR-14					
Name	Modify Worker					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to individually modify a worker from the whole set of workers if necessary.					

Table 19 - UR-14: Modify Worker

Identifier	UR-15					
Name	Delete Worker					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to delete and remove the workers required from the existing workers list					

Table 20 - UR-15: Delete Worker

Identifier	UR-16					
Name	View list of Workers					
Type	Capability		X	Constraint		
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The user will be able to view the list of potential workers available for the current project					

Table 21 - UR-16 View list of Workers

Identifier	UR-17
Name	Obtain results

Type	Capability		X	Constraint	
Priority	High	X	Medium		Low
Optionality	Mandatory		X	Optional	
Description	The user will be able to obtain the results for the team Optimization process for a given project				

Table 22 - UR-17 Obtain results

Identifier	UR-18				
Name	Export results				
Type	Capability		X	Constraint	
Priority	High	X	Medium		Low
Optionality	Mandatory			Optional	X
Description	The user will be able to export and save the results from the optimization process				

Table 23 - UR-18: Export results

8.1.2. Constraint Requirements

Constraint requirements define the restrictions that the user will find at the time of interacting with the system, and they may be related to the user interface interactions, the available resources, etc.

Identifier	UR-19				
Name	English language				
Type	Capability			Constraint	X
Priority	High	X	Medium		Low
Optionality	Mandatory		X	Optional	
Description	The user will visualize the program and all its contents in English.				

Table 24 - UR-19: English language

Identifier	UR-20					
Name	Dialog messages					
Type	Capability			Constraint		X
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	If some of the required fields are not fulfilled by the user, a dialog will be displayed in order to remember him to fulfill the missing data.					

Table 25 - UR-20: Dialog messages

Identifier	UR-21					
Name	Export results format					
Type	Capability			Constraint		X
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The results of the optimization will be exported in Excel format (.xls)					

Table 26 - UR-21: Export results format

Identifier	UR-22					
Name	Save data					
Type	Capability			Constraint		X
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The project information (along with the available workers) will be saved in the system database					

Table 27 - UR-22: Save data

Identifier	UR-23					
Name	Load data					
Type	Capability			Constraint		X
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	The project information (along with the available workers) will be loaded from the system database.					

Table 28 - UR-23: Load data

Identifier	UR-24					
Name	One active project					
Type	Capability			Constraint		X
Priority	High	X	Medium		Low	
Optionality	Mandatory		X	Optional		
Description	There can only be one active project at a time. If the user wants to work with other project the current one will be closed.					

Table 29 - UR-24: One active project

8.2. Use Cases

In this section, the different scenarios in which the end user can interact with the System will be defined. These scenarios are called Use Cases, and they represent the list of steps to be followed in order to achieve a particular goal.

The following figures represent the different Use Cases defined for this project; they are classified in four different groups so as to ease their visualization and understanding.

The first figure displays the Use Cases related to the registration and logging interaction, the second one defines the interactions related with the project management, the third figure contains the Use Cases for the interaction with workers and the last one contains the ones regarding the results of the optimization.

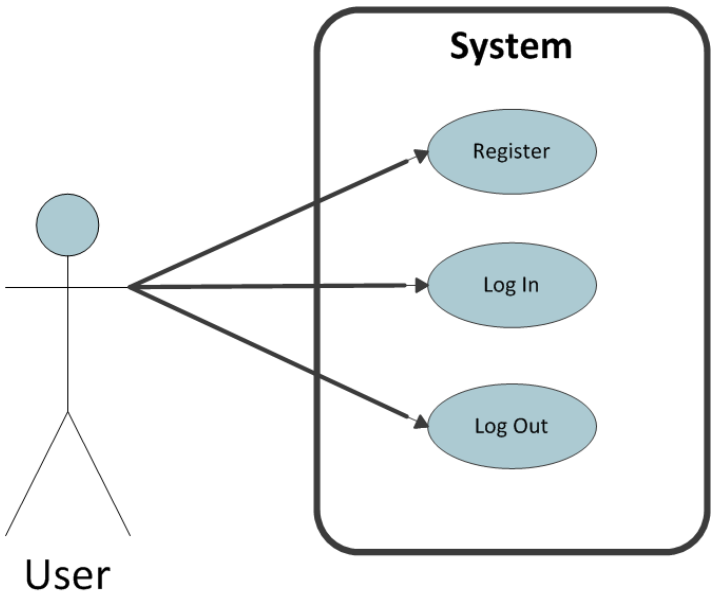


Figure 18 - Access Use Cases

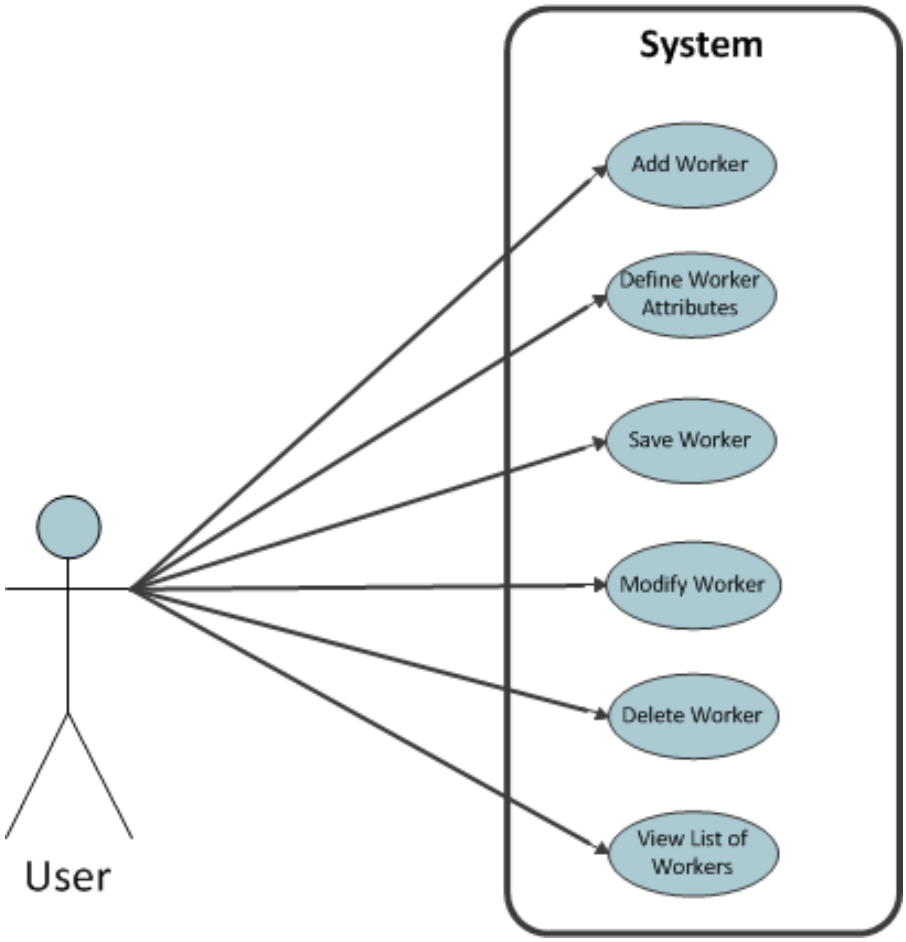


Figure 19 - Project Use Cases

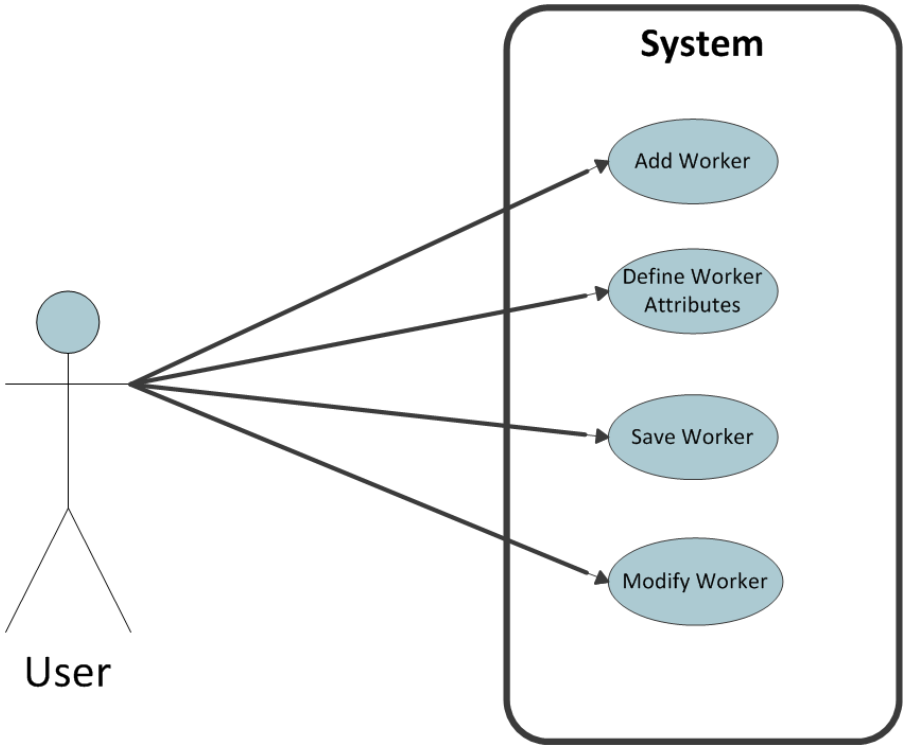


Figure 20 - Worker Use Cases

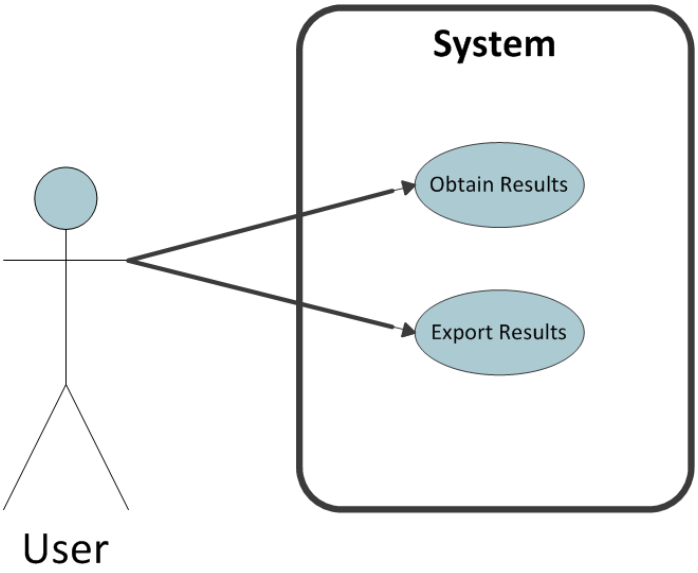


Figure 21 - Results Use Cases

After having a general view of which are the Use Cases considered, the following tables will provide a more extensive and concrete description of each of the Use Cases. The tables will provide information distributed in the followed fields:

- **Use Case ID**: Unique Identifier of the User Case with the following format "UC-XX", being XX a number that identifies the Use Case.
- **Use Case Name**: The name given to the Use Case.
- **Actors**: The ones executing the action (could be a person or a system).
- **Description**: A brief description of the Use Case scenario.
- **Pre-Conditions**: Conditions that must have been previously met so that the Use Case can take place.
- **Standard Flow**: The usual flow of steps followed in the Use Case from the beginning to the end.
- **Post-Conditions**: Condition situation after the Use Case takes place.

Use Case ID	UC-01
Use Case Name	**Register**
Actors	User
Description	The User registers into the system.
Pre-Conditions	The user does not have an account to access the system
Standard Flow	• The user clicks on the register button • The user types his username, e-mail and his password • The user confirms the introduced information and completes the registration process.
Post-Conditions	The user is now registered into the system with its username as ID and his password. The user email account and the username cannot be used by other users.

Table 30 - UC-01: Register

Use Case ID	UC-02
Use Case Name	**Log In**
Actors	User
Description	The user logs into the system
Pre-Conditions	The user must have previously registered into the system
Standard Flow	• The user introduces his username and password • The user clicks on the Log In button
Post-Conditions	The user can access now to all the system functionalities

Table 31 - UC-02: Log In

Use Case ID	UC-03
Use Case Name	**Log Out**
Actors	User
Description	The user logs out of the system
Pre-Conditions	The user must have previously logged into the system
Standard Flow	• The user clicks on the Log Out button
Post-Conditions	The Login/Register screen is displayed

Table 32 - UC-03: Log Out

Use Case ID	UC-04
Use Case Name	**Add a Project**
Actors	User
Description	Create and Add a new project
Pre-Conditions	The user must be logged into the system
StandardFlow	• The user clicks on the “Add Project” button
Post-Conditions	A new tab for defining the project’s attributes is open.

Table 33 - UC-04: Add a Project

Use Case ID	UC-05
Use Case Name	**Define Attributes**
Actors	User
Description	Defining the project's attributes
Pre-Conditions	The program must be in the correct tab for the attributes definition
Standard Flow	• The user gives a name to the project • The user gives a value to the project Size.
Post-Conditions	

Table 34 - UC-05: Define Attributes

Use Case ID	UC-06
Use Case Name	**Select a Methodology**
Actors	User
Description	Selecting a Methodology for the project
Pre-Conditions	The program must be in the correct tab for the attributes definition
Standard Flow	• The user gives user selects a methodology from the list of methodologies available
Post-Conditions	The selection of methodology will be taken into account

Table 35 - UC-06: Select a Methodology

Use Case ID	UC-07
Use Case Name	**Select an Attribute**
Actors	User
Description	Selecting a particular attribute important for the project
Pre-Conditions	The program must be in the correct tab for the attributes definition
Standard Flow	• The user gives user selects one attribute from the available ones
Post-Conditions	The highlight of the attribute will be taken into account

Table 36 – UC-07: Select an Attribute

Use Case ID	UC-08
Use Case Name	**Save Project**
Actors	User
Description	Save the current project attributes
Pre-Conditions	The name and the size of the project must be defined
Standard Flow	• The user clicks on the "Save Project" button • A window offers the user the possibility to save the project information
Post-Conditions	The new project is now created.

Table 37 - UC-08: Save Project

Use Case ID	UC-09
Use Case Name	**Load Project**
Actors	User
Description	The user loads the information from a previously created project
Pre-Conditions	The user must be logged into the system. The user must have a previously created project.
Standard Flow	• The user clicks on the Load Project button • The user selects the project he wants to load
Post-Conditions	The selected project is now available for use (along with the list of workers assigned to the project).

Table 38 - UC-09: Load Project

Use Case ID	UC-10
Use Case Name	**Delete Project**
Actors	User
Description	The user deletes the current project
Pre-Conditions	The project to delete must me open
Standard Flow	• The user clicks on the delete project button • The user confirms the action
Post-Conditions	The current project, along with its attributes and its workers are is deleted from the database and removed from the system.

Table 39 - UC-010: Delete Project

Use Case ID	UC-11
Use Case Name	**Add a Worker**
Actors	User
Description	The user adds a worker for the current project.
Pre-Conditions	The user must be logged into the system. The user must have previously created a project or loaded a project to be able to add workers to it.
Standard Flow	• The user goes to the Add Worker tab • The user clicks on the Add Worker button
Post-Conditions	The screen to define the worker attributes is now available

Table 40 - UC-11: Add a Worker

Use Case ID	UC-12
Use Case Name	**Define a Worker's Attributes**
Actors	User
Description	The user defines the attributes of a given worker
Pre-Conditions	The user must be in the Add Worker screen.
Standard Flow	• The user defines the name of the worker • The user defines the different attributes of the worker
Post-Conditions	

Table 41 - UC-12: Define a Worker's Attributes

Use Case ID	UC-13
Use Case Name	**Save a Worker**
Actors	User
Description	The user saves all the information of the new worker
Pre-Conditions	The user must be in the Add Worker screen. The user must have fulfilled all the required fields for the worker definition.
Standard Flow	• The user clicks on the Save Worker button
Post-Conditions	The new worker is saved and added to the list of workers for the current project.

Table 42 - UC-13: Save a Worker

Use Case ID	UC-14
Use Case Name	**Modify a Worker**
Actors	User
Description	The user modifies the attributes of a worker
Pre-Conditions	The user must have previously created the user. The user must be on the list of workers screen.
Standard Flow	• The user clicks on the Edit button on the list of workers corresponding to the desired worker • The user modifies the attributes • The user clicks on the Save Worker button
Post-Conditions	The worker information is now updated

Table 43 - UC-14: Modify a Worker

Use Case ID	UC-15
Use Case Name	**Delete a Worker**
Actors	User
Description	The user deletes a worker
Pre-Conditions	The user must have previously created the worker. The user must be on the list of workers screen.
Standard Flow	• The user clicks on the Delete button on the list of workers corresponding to the desired worker • The user confirms the operation
Post-Conditions	The selected worker is removed from the list of workers

Table 44 - UC-15: Delete a Worker

Use Case ID	UC-16
Use Case Name	**View the list of Workers**
Actors	User
Description	The user can see the list of workers for the current project
Pre-Conditions	There must be workers in the list
Standard Flow	• The user clicks on the list of workers tab
Post-Conditions	The list of Workers along with the edition and deletion options are displayed

Table 45 - UC-16: View the list of Workers

Use Case ID	UC-17
Use Case Name	**Obtain Results**
Actors	User
Description	The user obtains the results of the optimization process for the current project
Pre-Conditions	The project must be created. There must be workers added to the project.
Standard Flow	• The user clicks on the Optimizer tab • The user clicks on the Obtain results button
Post-Conditions	The results of the optimization are displayed on screen to the user

Table 46 - UC-17: Obtain Results

Use Case ID	UC-18
Use Case Name	**Export Results**
Actors	User
Description	The user exports the results of the optimization in Excel format.
Pre-Conditions	The optimization process must have been carried out previously
Standard Flow	• The user clicks on the Export Results button • The user save the file in his computer
Post-Conditions	

Table 47 - UC-18: Export Results

8.3. Software Requirements

Software requirements are inferred from the previously defined User Requirements, and they are used to define what the system must do in order to satisfy the needs of the end users. They are defined in an abstract way avoiding technical details so as to provide requirements which are easily understood.

In the next subsections functional and non-functional Software Requirements are going to be stated and described. The fields which are going to be defined for every Software Requirement are the following:

- **Identifier**: Unique ID of the requirement with the format "SR-XX", where XX determines the number of requirement.
- **Name**: Descriptive name of the requirement
- **Type**: Type of User Requirement (Functional or Non-Functional)
- **Priority**: Priority of the requirement being fulfilled (High, Medium or Low)
- **Optionality**: This field defines if the requirement is optional or not.
- **Description**: Brief description of the requirement
- **Source**: User Requirements which inferred the Software Requirement

8.3.1. Functional Requirements

Functional requirements are used in order to define what the system must do, what it should accomplish to, but not how to do it. They establish the different functionalities that the system must have and by that, they define the behavior of the system in different situations.

Identifier	SR-01					
Name	User Registration					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to register and to create an account with their email, a username and a password.					
Source	UR-01					

Table 48 - SR-01: User Registration

Identifier	SR-02					
Name	User Access					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to access the system by using his username and a password					
Source	UR-02					

Table 49 - SR-02 User Access

Identifier	SR-03					
Name	User Exit					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to exit the system					
Source	UR-03					

Table 50 - SR-03: User Exit

Identifier	SR-04					
Name	Project Creation					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the creation of Software projects					
Source	UR-04					

Table 51 - SR-04: Project Creation

Identifier	SR-05					
Name	Project’s Name					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**		**Medium**	X	**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the definition of the project’s name					
Source	UR-05					

Table 52 - SR-05: Project’s Name

Identifier	SR-06					
Name	Project’s Attributes					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the definition of the project size					
Source	UR-05					

Table 53 - SR-06: Project's Attributes

Identifier	SR-07					
Name	Project’s Methodology					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the selection of a Methodology for the optimization					
Source	UR-06					

Table 54 - SR-07: Project's Methodology

Identifier	SR-08					
Name	Project’s Essential Attribute					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow to highlight a particular worker’s attribute for the optimization					
Source	UR-07					

Table 55 - SR-08: Project's Essential Attribute

Identifier	SR-09					
Name	Save Project					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to save the information of the current project					
Source	UR-08					

Table 56 - SR-09: Save Project

Identifier	SR-10					
Name	Load Project					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to load a previously saved project from the database					
Source	UR-09					

Table 57 - SR-10: Load Project

Identifier	SR-11					
Name	Delete Project					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**		**Medium**	X	**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to delete an existing project from the database					
Source	UR-10					

Table 58 - SR-11: Delete Project

Identifier	SR-12					
Name	Add Worker					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to add a new potential worker to the current project					
Source	UR-11					

Table 59 - SR-12: Add Worker

Identifier	SR-13					
Name	Worker's Name					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**		**Medium**	X	**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to specify the name of the new worker					
Source	UR-12					

Table 60 - SR-11: Worker's Name

Identifier	SR-14					
Name	Worker's Attributes					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to specify the different attributes of the new worker					
Source	UR-12					

Table 61 - SR-14: Worker's Attributes

Identifier	SR-15					
Name	Save Worker					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to save the new worker					
Source	UR-13					

Table 62 - SR-15: Save Worker

Identifier	SR-16					
Name	Modify Worker					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to modify the attributes of a previously created worker					
Source	UR-14					

Table 63 - SR-16: Modify Worker

Identifier	SR-17					
Name	Delete Worker					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to delete a previously created worker from the list of workers of the project					
Source	UR-15					

Table 64 - SR-17: Delete Worker

Identifier	SR-18					
Name	List of Workers					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to view the list of Workers that have been added to the current project.					
Source	UR-16					

Table 65 - SR-18: List of Workers

Identifier	SR-19					
Name	Obtain Results					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user generate the optimal combination of Workers for the current project					
Source	UR-17					

Table 66 - SR-19: Obtain Results

Identifier	SR-20					
Name	Export Results					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**		**Medium**	X	**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will allow the user to export the results obtained from the optimization process					
Source	UR-18					

Table 67 - SR-20: Export Results

Identifier	SR-21					
Name	Access to the Database					
Type	**Functional**		X	**Non-Functional**		
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will be able to communicate with the database in order to store and retrieve information from it.					
Source	UR-01, UR-02, UR-03, UR-04, UR-05, UR-06, UR-07, UR-08, UR-09, UR-10, UR-11, UR-12, UR-13, UR-14, UR-15, UR-16, UR-17, UR-18					

Table 68 - SR-21: Access to the Database

8.3.2. Non-Functional Requirements

Non-Functional Requirements define the limitations and constraints of the system, without providing any functionality, and they are defined from the Constraint User Requirements.

Identifier	SR-22					
Name	English Language					
Type	**Functional**			**Non-Functional**		X
Priority	**High**	X	**Medium**		**Low**	
Optionality	**Mandatory**		X	**Optional**		
Description	The system will provide the information in English					
Source	UR-19					

Table 69 - SR-22: English Language

Identifier	SR-23					
Name	Dialog Messages for Project					
Type	Functional			Non-Functional		X
Priority	High	X	Medium		Low	
Optionality	Mandatory	X	Optional			
Description	The system will display a dialog message in case some of the fields required in the project creation is not fulfilled					
Source	UR-20					

Table 70 - SR-23: Dialog Messages for Project

Identifier	SR-24					
Name	Dialog Messages for Worker					
Type	Functional			Non-Functional		X
Priority	High	X	Medium		Low	
Optionality	Mandatory	X	Optional			
Description	The system will display a dialog message in case some of the fields required in the worker creation is not fulfilled					
Source	UR-21					

Table 71 - SR-24: Dialog Messages for Worker

Identifier	SR-25					
Name	Results Format					
Type	Functional			Non-Functional		X
Priority	High	X	Medium		Low	
Optionality	Mandatory	X	Optional			
Description	The system will export the results of the optimization in .xls format					
Source	UR-22					

Table 72 - SR-25: Results Format

Identifier	SR-26					
Name	Save Data					
Type	Functional			Non-Functional		X
Priority	High	X	Medium		Low	
Optionality	Mandatory	X		Optional		
Description	The system will save and store the project information (with its workers information) in the database					
Source	UR-23					

Table 73 - SR-26: Save Data

Identifier	SR-27					
Name	Load data					
Type	Functional			Non-Functional		X
Priority	High	X	Medium		Low	
Optionality	Mandatory	X		Optional		
Description	The system will load the saved projects information from the database (as well as the workers for that project)					
Source	UR-24					

Table 74 - SR-27: Load data

Identifier	SR-28					
Name	One active project					
Type	Functional			Non-Functional		X
Priority	High	X	Medium		Low	
Optionality	Mandatory	X		Optional		
Description	The system will allow only one active project at the same time					
Source	UR-25					

Table 75 - SR-28: One active project

8.4. Traceability Matrix

The following tables are Traceability Matrixes and they are used in order to see the correlations existing among the User Requirements and the System Requirements. They are also useful at the time of checking if the currents requirements are being met and if for each User Requirements there is at least one System Requirement to fulfill it.

In the first table we can see the Traceability Matrix which relates the Capability User Requirements and the Functional System Requirements and in the second one the Constraints User Requirements are checked against the Non-Functional System Requirements.

It can be observed that most of the times for each User Requirements there is a System Requirement related to it, and sometimes two, so by checking these Matrixes we can be sure that the User Requirements are being properly collected in the System Requirements.

	UR-01	UR-02	UR-03	UR-04	UR-05	UR-06	UR-07	UR-08	UR-09	UR-10	UR-11	UR-12	UR-13	UR-14	UR-15	UR-16	UR-17	UR18
SR-01	X																	
SR-02		X																
SR-03			X															
SR-04				X														
SR-05					X													
SR-06					X													
SR-07						X												
SR-08							X											
SR-09								X										
SR-10									X									
SR-11										X								
SR-12											X							
SR-13												X						
SR-14												X						
SR-15													X					
SR-16														X				
SR-17															X			
SR-18																X		
SR-19																	X	
SR-20																		X
SR-21	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 76 - Traceability Matrix: Functional vs. Capability Requirements

	UR-19	UR-20	UR-21	UR-22	UR-23	UR-24
SR-22	X					
SR-23		X				
SR-24		X				
SR-25			X			
SR-26				X		
SR-27					X	
SR-28						X

Table 77 - Traceability Matrix: Non-Functional vs. Constraint Requirements

PART IV:

ARCHITECTURE

Chapter 9 | Design of the Architecture

9.1. Introduction

The Software Architecture of a system represents and describes the set of components and modules which compose it and the relationships among these different structures.

The Architecture is useful in a way that it provides a general and abstract vision of the whole system and the different interactions between its components, making easier the understanding and the workflow of the system as a whole.

In order to design the architecture of a system, there are several models to be used and followed, allowing different approaches for the design of the system distribution. The one which was found most suitable for this project is the Three-tier Architecture, which presents three different layers for presentation, logic and data management; in the following section this decision will be explained as well as the functioning of the architecture.

9.2. Three-tier Architecture

The chosen Architecture for this particular project has been the Three-tier Architecture, a derivation of the Multi-tier Architecture. In these kinds of Architectures, the functionalities for presentation, logic processing and data management are separated modules and independent from each other.

The three different tiers or layers that this Architecture provides are the following ones:

- **Presentation Layer**: This tier is at the top level of the Architecture and is the one in charge of displaying all the application level information to the end user through a graphic interface. This layer communicates with the Business Layer to obtain and send information to it.

- **Business Layer**: This tier is in charge of controlling the application functionality and all the logical processes and calculations behind the Presentation Layer. The Presentation Tier provides the Business Layer with input data, and this one is in charge of processing it in the proper way and sending it back to the User Interface

- **Data Layer**: The third layer is in charge of managing all the information that is stored and requested from the database. The information stored in this layer is independent of the other two tiers.

The following image provides a schematic representation of the communication flow between the three different and independent layers (and the database) and how they are spatially distributed.

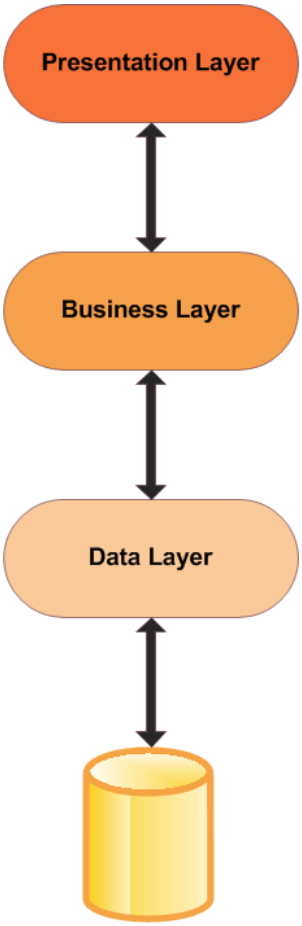


Figure 22 - Three-tier Architecture

9.3. Architecture Definition

The following figure provides a general vision of the System's Architecture in which we can see the three differentiated layers that have been explained previously.

Each of the three layers contains their own classes which are able of communicating with each other in order to make the whole program work, but maintaining their field of operation depending on the layer they belong to.

The Presentation Layer classes will have functionalities and attributes related to the management of the information display and the user interface, while the ones in the Business Layer will be in charge of doing all the calculations, information processing and heavy and the Data Layer classes will have to store all the information introduced through the User Interface and processed in the Business Layer, all of these working independently from each other in their own layers.

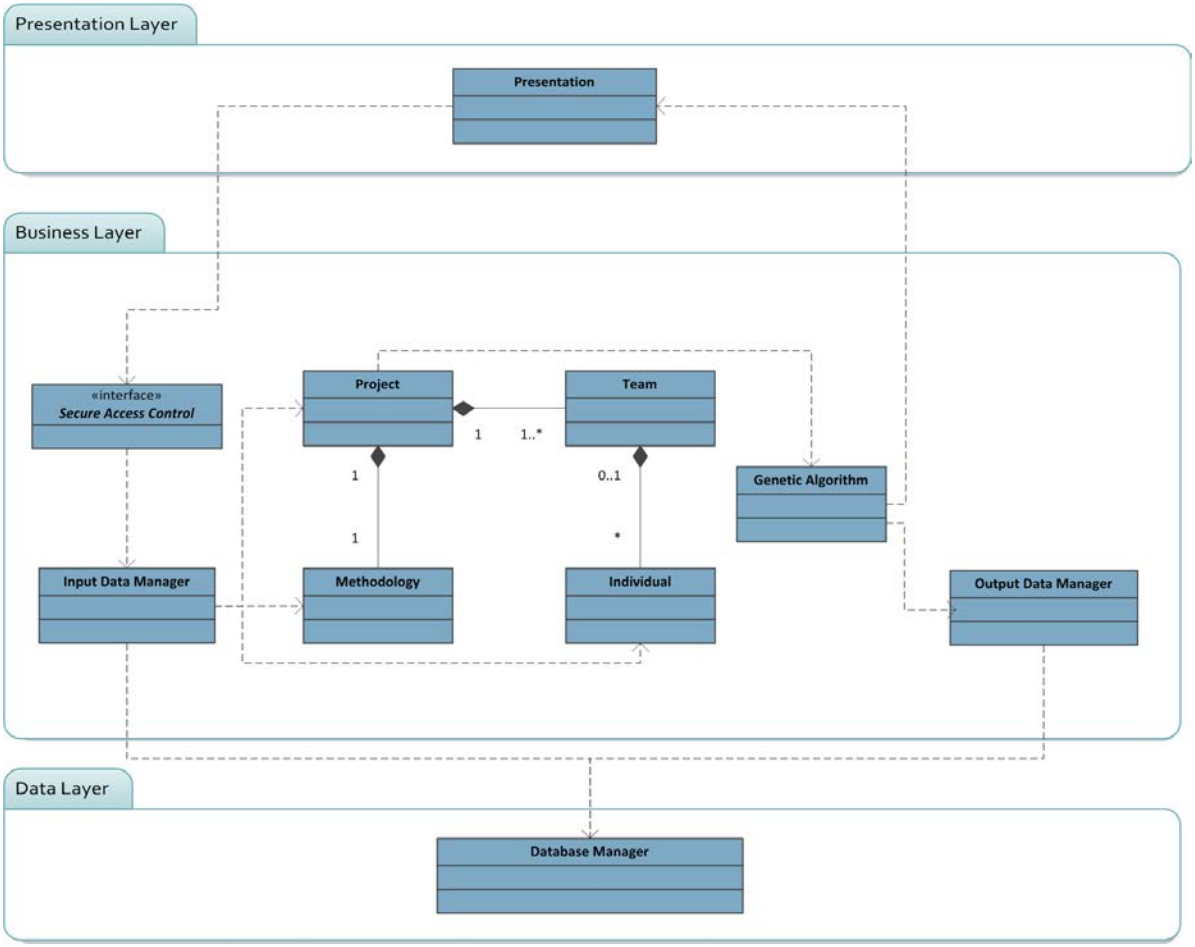


Figure 23 - System's Architecture

Chapter 10 | Class Definition

10.1. Presentation Layer

The Presentation Layer is in only in charge of the User Interface handling, this means, handling the communication and the interactions with the end user.

This layer's objective is to receive the input data introduced by the user and to pass it to the Business Layer for its processing; after the required calculations, this layer will be the one to return the end user with the resultant information from a particular request the user makes.

The class in charge of handling all the Graphic User Interface Communication is the Presentation Class and it will be explained bellow.

10.1.1. Presentation

The Presentation is a simple class responsible of the information displayed on the screen and the direct communication with the user.

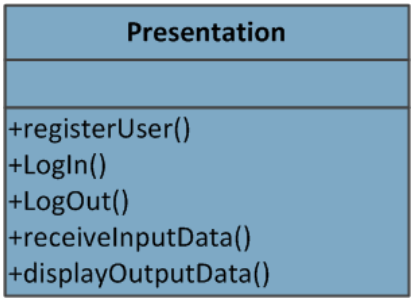


Figure 24 - Presentation Class

The methods that help this class to carry through their presentation duties are the following:

- **registerUser():** displays on and manages the Registration screen.
- **LogIn()**: displays and manages the Log In screen.
- **LogOut()**: manages the Log Out of the user
- **receiveInputData()**: manages the incoming input data introduced by the user and sends it to the Business Layer.
- **displayOutputData()**: receives output information from the Business Layer and displays it on screen to the user

10.2. Business Layer

The Business Layer contains the majority of the classes which compose the system and it is where most of the heavy and critical work is done.

In this layer, the access of the user is controlled, the input data is processed and parsed and the Genetic Algorithm is able to process the information and then send back the results to the presentation Layer for displaying and to the Data Layer for storage.

10.2.1. Secure Access Control

The first class belonging to the Business Layer is the Secure Access Control class, in which, as its name claims, the access of the different users into the system is controlled.

Figure 25 - Secure Access Control Class

Basically, the objective of this class is controlling and handling the authentication of users, in order to grant them access to the specific functionalities of the system further the login screen.

In case the authentication process is not properly fulfilled (for example, incorrect login), the current user will not be allowed to access into the system and therefore, he will not be able to generate optimal working teams. The method by which this class manages the accesses to the system is:

- **controlAccess()**: manages the access of the users into the system. For a user to be able to access the system he must be registered and the tuple Username-Password must match with the registration data stored in the database

10.2.2. Input Data Manager

The Input Data Manager class is used in order to manage all the data introduced by the user through the User Interface; the Presentation Layer send this information to this

class after the authentication process and this class is in charge of parsing and processing all these input information in a way that the rest of the classes will be able to work with the user information.

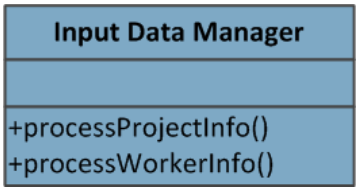


Figure 26 - Input Data Manager Class

The methods contained in this class are listed below:

- **processProjectInfo()**: gets the project information introduced by the user in the Presentation Layer and parses it into the proper format so the Genetic Algorithm can work with that information.
- **processWorkerInfo()**: gets the worker information introduced by the user in the Presentation Layer and parses it into the proper format so the Genetic Algorithm can work with that information.

10.2.3. Project

The project class is used in order to store all the information related to the project and it is the main class through where the whole program is called.

After the completion of the optimization process, the different fields of the project class are fulfilled with the results of the Genetic Algorithm executions and this information is further sent to the User Interface for the users to observe the results of the optimal team for his project.

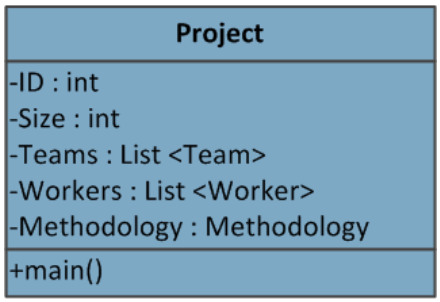


Figure 27 - Project Class

10.2.4. Methodology

The methodology class containers the previously explained attributes related to the methodology and two simple methods which are used in order to get and set the most appropriate methodology for the current project.

- **getMethodology()**: gets the value of the Methodology Object
- **setMethodology()**: sets a Methodology Object to some value
- **Methodology():** Constructor for the Methodology Object

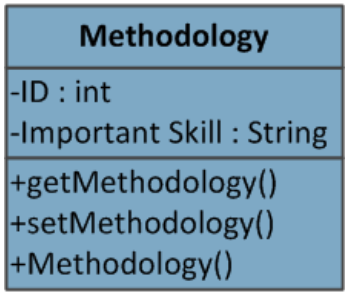


Figure 28 – Methodology Class

10.2.5. Worker

The Worker class contains all the different capabilities of a worker which are considered at the time of calculating the fitness function in the Genetic Algorithm; all these competences have been previously explained in the analysis section of this document.

The class contains a set of methods which are mainly used of initialization of the worker itself, and to get and set the values of the Unique Identifier of the worker and the different set of skills and abilities which will vary considerably between different workers.

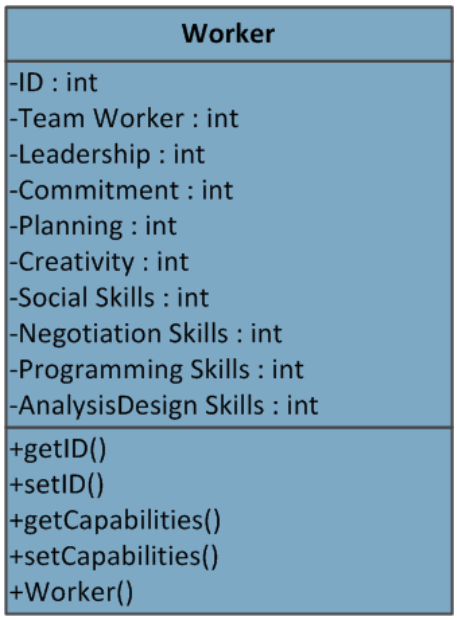


Figure 29 - Worker Class

In the program, there is a get and set methods for each different capability of the worker, in order to simplify and avoid redundancy, in this document they have been summarized together into getCapabilities() and setCapabilities().

- **getID()**: gets the unique ID of a Worker Object.
- **setID()**: gives value to the unique ID of a worker when it is created.
- **getCapabilities()**: gets the values of all the different capabilities of a worker.
- **setCapabilities()**: sets the all the capabilities of a worker.
- **Worker()**: Constructor for the Worker Object.

10.2.6. Team

The Team class contains the information related to each team that is considered for being the chosen one for the current project.
The class contains the usual getters and setters, a constructor and an auxiliary method which will help the program at the time of comparing to teams in order to prevent repeated team configurations.

- **getID()**: gets the value of the unique ID of a team.
- **setID()**:The way of uniquely identifying a team is by creating an integer array, in which each position of the array contains a worker, and by ordering those workers' IDs, we can assure that the team is unique and is easy to identify so as not to repeat the same team configuration.

- **getWorkers()**: gets the list of workers belonging to the team.
- **setWorkers()**: defines the group of workers which are going to be part of the team.
- **getFitness()**: gets the result of the Fitness evaluation of the team.
- **setFitness()**: sets the Fitness value for the team by using the Fitness function.
- **equals()**: This method compares two teams ID attribute and determines if both team configurations are equal or not, in order to avoid repeated teams.
- **Team():** Constructor for the team object

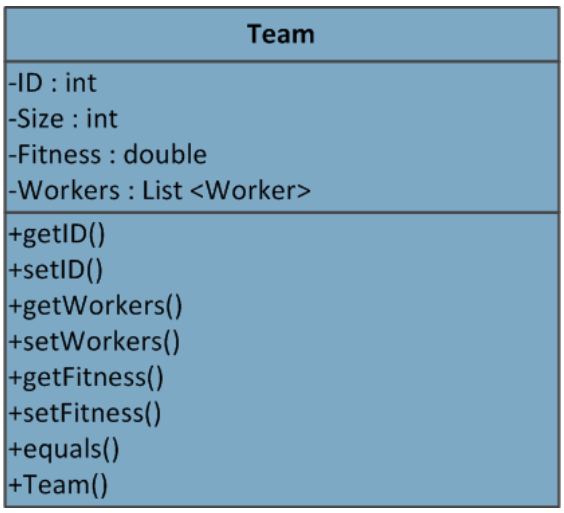


Figure 30 - Team Class

10.2.7. Genetic Algorithm

The Genetic Algorithm class is the core engine of the whole system where most of the logic and calculations are made as it is in charge of the whole evaluation and optimization process of the working teams.

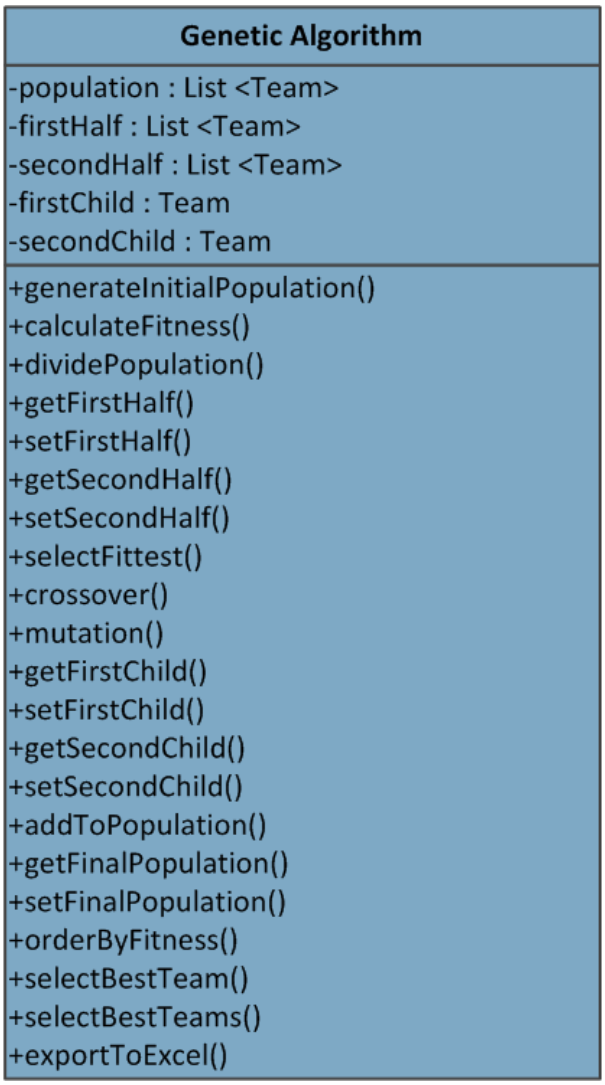


Figure 31 - Genetic Algorithm Class

As its attributes, the class contains the whole population (the different teams configured) and then the first and second half of teams (generated randomly) which are extracted from the complete population list and will be used to bring some randomness to the selection process and finally, the first and the second children generated after the reproductive process which will be part of the next generation of the population. The different functions that this class uses to manage all the reproductive and evolutionary process will be listed and described below.

The other main objective of this class is to process and organize all the data previously generated by the GA calculations in order to be able to provide the end user with specific and concise information on the optimization.

For this purpose, the class obtains the final population resultant and orders the teams by the value of their fitness function. After that, it offers the user the possibility of displaying the best team generated according to fitness or the 5 best team combinations generated in that execution of the program. It is in charge of sending back the results of the execution to the Presentation Layer, so the Presentation Class will be able to display the results of the optimization to the end user and also provides a functionality which allows the user to export the results of the optimization.

The different functions that make this possible are:

- **generateInitialPopulation()**: generates a random initial population of teams
- **calculateFitness()**: evaluates the Fitness of every individual included in the population taking into account all the capabilities of the different workers in each team.
- **dividePopulation()**: divides the existing population in two halves which are randomly chosen.
- **getFirstHalf()/getSecondHalf():** get the team Objects belonging to the first or the second half in which the population is divided.
- **setFirstHalf()/setSecondHalf():** sets a team to be in the first half of the population or in the second one.
- **selectFittest()**: selects the fittest individual from each half of the population, the one with the higher Fitness value.
- **crossover()**: crosses the genes from the chosen parents in order to generate two children.
- **mutation()**: mutates with a probability of 0.1 the genes of a offspring. This has to be done over the two offspring separately.
- **getFirstChild()/getSecondChild():** gets the team workers from each of the new children.
- **setFirstChild()/setSecondChild():** sets the team workers of each of the new children
- **addToPopulation():** adds the new children to the current population except when the generated offspring already exist in the population.
- **getFinalPopulation():** gets all the team individuals belonging to the final population
- **setFinalPopulation():** sets all the individuals which will be part of the final population
- **orderByFitness():** orders all the individuals belonging to the final population according to their fitness value

- **selectBestTeam():** selects the team which has the best fitness value (the first of the list)
- **selectBestTeams():** selects the 5 best teams with the highest fitness values (the first 5 teams of the list)
- **exportToExcel():** exports the results of the optimization to .xls format.

10.2.8. Output Data Manager

The Output Data Manager's main functionality is the communication with the Data Layer, in which all the required information will be saved and stored for future usage. It has only one method:

- **saveData()**: sends all the information to the Data Layer so it can be stored in the database for future use.

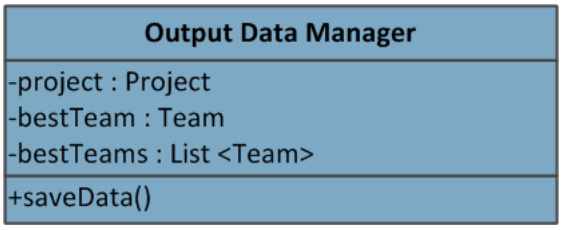


Figure 32 - Output Data Manager Class

10.3. Data Layer

The Data Layer only contains a class, the Database Manager and it is change of communicating with the Business Layer in order to request and send the required information from and to it and it is also responsible for the communication with the database in which all the critical information about projects and workers is stored.

10.3.1. Database Manager

The Database Manger class is in charge of managing all the information that comes in or out the Database; its objective is to store and save the new information which has been previously introduced by the user as well as is in charge of sending the application the information which has already been stored and has been requested again by the user through the User Interface.

- **addData()**: adds new information to the database.
- **editData()**: edits existing information from the database.

- **deleteData()**: deletes existing data from the database.
- **getData()**: retrieves the requested data from the database and sends it back to the Business and Presentation Layers for processing.

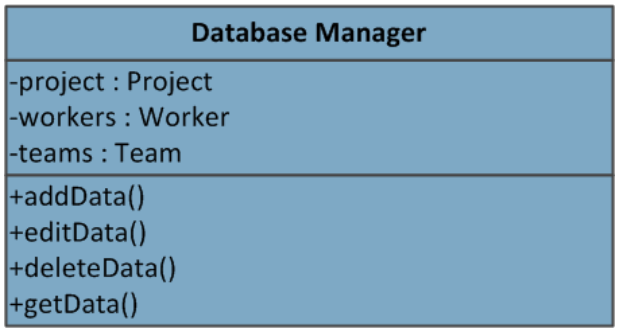


Figure 33- Database Manager Class

PART V:

DESCRIPTION OF THE SOLUTION

Chapter 11 | Overall Design Scheme of the Solution

In this chapter, the chosen solution for solving the working teams' optimization problem will be explained in full detail. The main objective of this project is the optimization of working teams, this means, obtaining the best team or teams which will be in charge of the development of a proposed project.

In order to achieve that goal, the chosen mechanism to get the best working team has been Genetic Algorithms, one of the reasons they were chosen, is because they are able of searching for multiple objectives at the same time, balancing them into the best results the algorithm allows.

The following sections will be distributed the following way; firstly, a general glance of the chosen solution, its workflow and its design will be provided; following, a justification of the chosen solution will be provided, as well as a discussion on the discarded solutions.

After that introduction to the solution scheme and the reasons why it has been chosen, an extensive walkthrough to the system's operation going through all the different phases which make the optimization of working teams possible.

11.1. Proposed Solution

The aim of this project is the optimization of software development working teams, for this purpose, a Genetic Algorithm based system has been designed and developed. In order to fully understand how the system works it is necessary to start from the basics; in the figure bellow, the basic workflow of the proposed system can be observed:

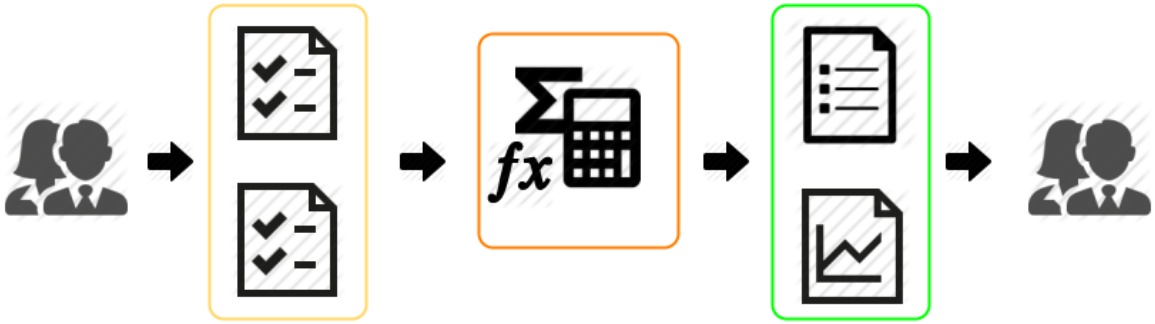


Figure 34 - System's workflow

Figure 31 defines in an abstract way the steps that are followed in order to obtain the most optimal team as possible:

- Firstly, the end user introduces the required information into the system (project, workers, etc.)
- After that, the system processes all the information manually added by the user and works on the optimal solution for the current problem.
- Finally, the system returns to the user with a set of optimal solutions which will help the end user at the time of designing working teams.

This process mentioned above can be repeated as much as the user wants, in case he wants to explore more possible solutions to the problem, or change some of the established parameters in order to see if the results obtained actually match with the optimal solution.

In a general way, the operation workflow of the system can be easily understood, but it is important to know what really happens inside the system, which are the calculations and processes that lead the system to provide one solution or another to the end user.

Figure 32 provides a summary of the behavior of the GA (a simplified version of the *Canonical GA*) used in the development of this project. In this algorithm, an initial random population of teams is created, immediately after that, it checks if the termination condition has been reached, in order to finish; in case not, the reproductive process starts.

The whole population is evaluated with the fitness function, after that, the fittest individuals are selection for crossover and generating two offspring, which may or not be affected by a mutation. In order to add these new individuals to the population, the only check made is that they do not already exist in the population and then start over again until the termination condition has been reached.

The way all these inner process related to the GA have been specifically defined will be explained in further detail along the following sections for a deep and precise understanding of every component.

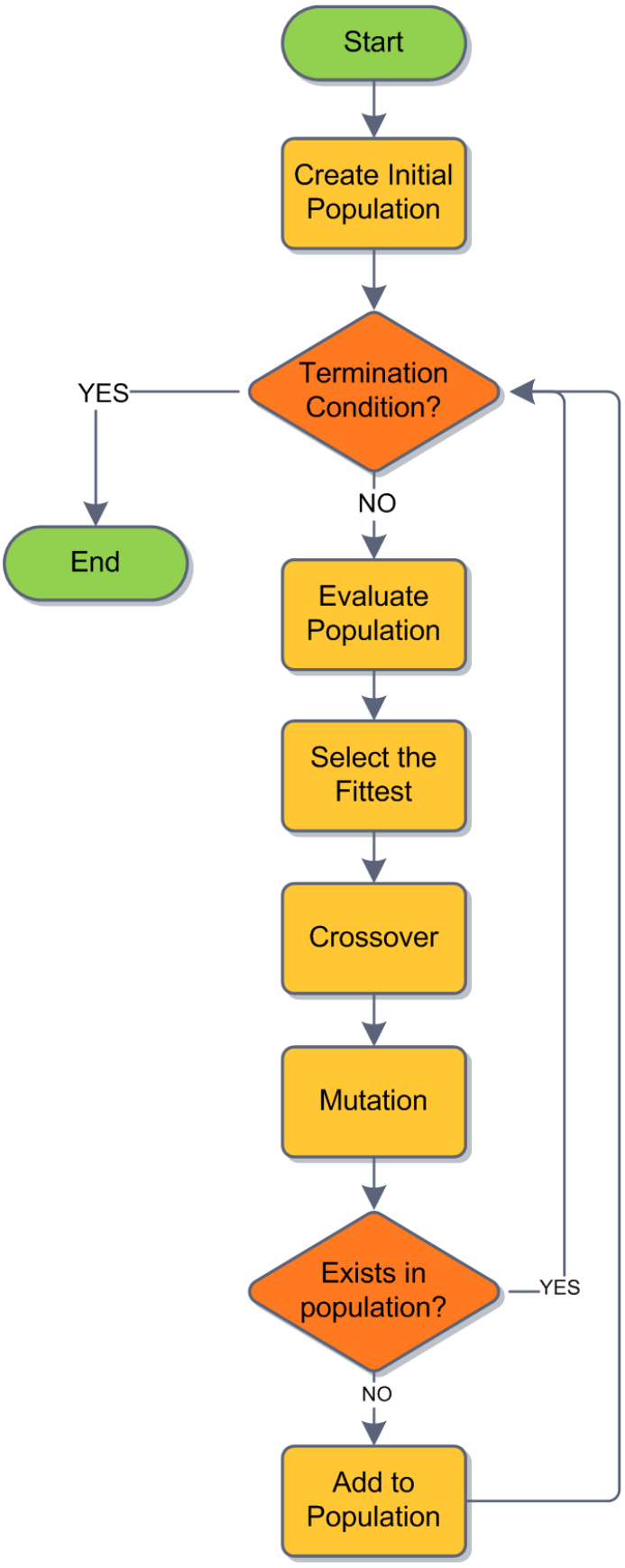


Figure 35 - Flow Chart of the GA

11.2. Justification of the solution

As it was said in previous sections, the selected choice for the design and implementation of this system in charge of optimizing working teams are the Genetic Algorithms.

The problem of designing optimal teams of any nature has always been present, and there are several and very diverse ways of confronting this problem and trying to reach an optimal solution, so the range of possibilities is huge for this particular problem.

Genetic Algorithms were chosen for this particular problem because of their flexibility, and the wide variety of options which allow refining and adjusting the different parameters so as to obtain the closest solution to the desired optimal solution as possible.

By using genetic algorithm, we are trusting in the natural selection, on the survival of the fittest, just like it happens in the real world, by properly evaluating the fitness of the individual of the populations we can manage the Algorithm in properly evolving the population and converging into the most optimal solution for the problem. The individuals which are fittest, and better adapt to the current environment (project) are the ones which are going to survive (be chosen) and generate further populations with their genes.

Other main reason for choosing GAs is that they allow multi-objective optimization; they are able of focusing on different objectives at the same time. In this project, we require the optimization of several capabilities, of several team workers at the same time, having two dimensions of optimization at the same time.

Genetic Algorithms can provide not only one solution, but a set of the best solutions obtained in one execution, so by a series of several executions the user is able of comparing the different results obtained and determining which are the fittest teams selected by the algorithm and then, choose the most suitable one for the current project.

The greater disadvantage about using Genetic Algorithms for optimization problems is that they are not able to assure an optimal solution. Because of the randomness that most of the GA stages involve, it is possible that the fittest individuals are not always the first choice and some other individuals may be the chosen ones. This problem can

be diminished by carefully defining the fitness function, and by setting a number of iterations large enough so that it will allow the fittest individuals to stand out among the not so fit and be the elected ones for the process.

11.3. Development of the Genetic Algorithm

In this section all the design decision regarding the Genetic Algorithm in the project will be explained in detailed, justifying the choices made for each of the GA different phases.

Inside the Selection section, the different choices of execution which will be available for the end user will be explained, as well as the different focus that the fitness function must have for each different use case.

11.3.1. Encoding

Before getting to the configuration of the GA itself, it is necessary to clearly define the structure that it has to have the data we are going to process in the Genetic Algorithm.

In this problem there are two main data structures which are going to be the ones taken into account in order to obtain a solution.

The individuals or chromosomes which are going to be evaluated by the Genetic Algorithm are the working teams, which in turn are composed by several workers; these ones would be the genes. Depending on the team size specified by the user, the “team” individual will have a different amount of genes composing it.

For example, this would be the codification for a chromosome representing a team formed by 5 workers with their IDs being 10, 50, 43, 15 and 25.

Figure 36 - Encoded team

In order to make it easier to identify and compare groups, the teams will have their genes ordered by the ID of the worker, and so we can make sure that no team composition is being repeated. The previous team configuration would look like this:

Figure 37 - Encoded and Ordered team

The following figure gives a graphical description of the distribution of a team and its different components for better understanding of the concept:

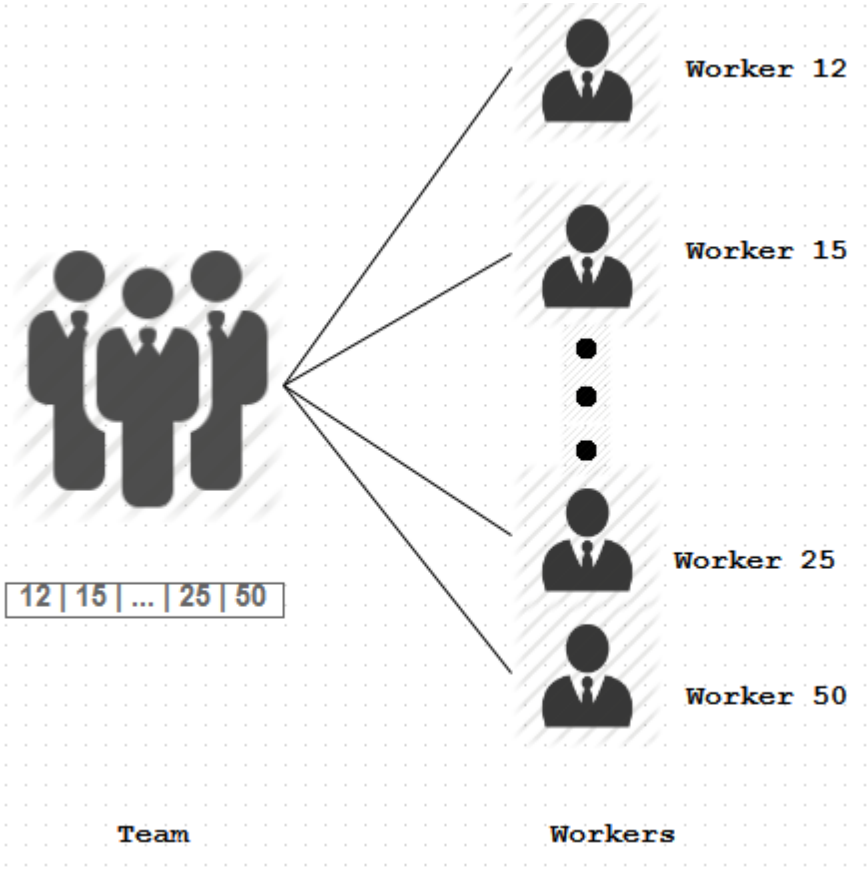


Figure 38 - Team and Workers

If we deepen into the configuration of the teams, it is necessary to know the information that each different gene (worker) provides to the final configuration of the team chromosome for its further evaluation using the fitness function.

The worker is identified by his unique ID, and it is not possible that two different workers share the same ID. Each of the workers will have their own configuration based on their particular skills and capabilities. Each of the positions defining a user will represent a different professional or personal skill, and the value they display is the score obtained by the user for that particular ability.

The possible values for this "skill" fields go from 0 to 10, being 0 the lowest rate and 10 the highest punctuation available for any competence. This is important because at the end, the skills of the worker are going to be the ones to determine in which team the user should belong (if any).

The following figure provides a visual definition of a worker and the different competences and skills which define him, including its ID which is used only for identification process and is not taken into account in further calculations.

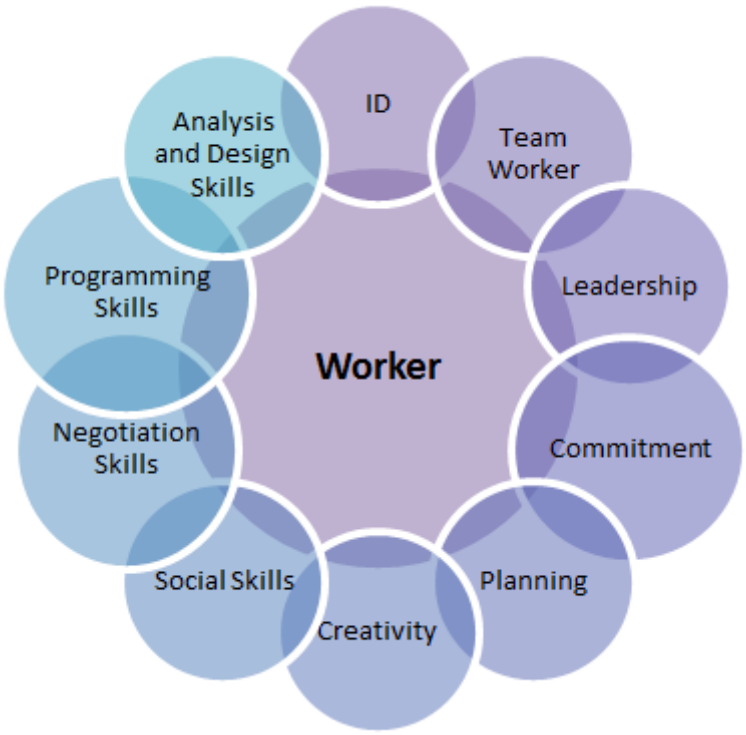


Figure 39 - Worker attributes

In the image below a complete user with its configuration fulfilled can be observed. The worker ID is 12, so there is no other worker with the same ID as this one which can be used in the team configurations.

On the right side, the attributes shown before can be seen, as well as the punctuations of each one for this particular that vary depending on the worker's abilities. In order to make it more visual and easier to understand, the capabilities which have high scores are colored in green, indicating that it is a positive value; the ones on average are yellow and the ones with non-desirable values are colored in red, indicating that these values are not desirable and should be "penalized"

Figure 40 - Worker configuration example

11.3.2. Initial Population

As it was explained before, the individuals which will form the population are the teams, which at the same time are composed by several workers each of them with their own configuration.

Once we know what will compose our population, the next step is to determine how this population is going to be generated, and its size.

For this particular problem it has been decided to generate an initial population whose size will be the number of available workers multiplied by two; this means, if I have 50 employees, the initial population will consists of 100 teams. These teams will be created randomly, having no specific criteria at the time of including the workers together in the different teams.

11.3.3. Evaluate Fitness

Once the initial Population has been randomly generated, the next step to take the evaluation of the Fitness of the different teams and with the help of that evaluation, determining which ones are the most adequate to enter the reproductive cycle.

This system provides the user with different execution modalities so as to find the optimal team according some set of needs. There is a basic operating mode and the following modes are derivations and modification of this base model.

Operation Mode 1

The first operation mode consists on obtaining the optimal team configuration from a set of workers given by the end user, with no more external parameters to be considered.

In order to evaluate the fitness of a team, it is also necessary to consider and evaluate the different capabilities of each of the workers on the team. This evaluation will be done with the help of the following equation which evaluates the capabilities of de different team members at the same time:

$$F = \sum_{n=1}^{j} \left(\frac{\sum_{n=1}^{i} \left(c_{ij} * w\left(c_{ij}\right) \right)}{i} \right) - f(l_i)$$

Equation 1- Fitness Function

- ***j*** = Total number of capabilities

- $\boldsymbol{c_{ij}}$ = Value of capability j for worker i
- $\boldsymbol{i}$ = Number of workers in the team
- $\boldsymbol{w(c_{ij})}$ = Function that multiplies the value of the current capability by a given weigh which will depend on the value of that capability for the user being evaluated.
- $\boldsymbol{l_i}$= Leadership capability value for worker i.
- $\boldsymbol{f(l_i)}$ = Function that evaluates de the leadership capability of the workers and penalizes if there are many members with high leadership value.

In order to be able to select the fittest team from a sample of randomized teams, it is necessary to evaluate them in relation to the score they have on each of the capabilities considering all the team members, encouraging high scores and punishing the lower values, so a capability with high score brings more value to the total value of the fitness evaluation than one with a lower value.

All the other worker's capabilities are considered "regular" capabilities, this means, they are capabilities which are always considered positive and desirable in a worker in which, the higher the score the better for the whole team.

Leadership has been considered also outside the main part of the equation, this has been done because leadership is a complicated capability which requires of a different analysis from the others. It is advisable that a team has a leader, but not so much if there is more than one person with high leadership skills in the same team because their characters are likely to crash and affect the whole team in the process. Later on the way proposed to evaluate this capability is described in detail

In the following equation, the different values that the function f(c) can take depending on the score of the capability are listed. It can be observed that, the higher the mark of the capability, the more the capability retains its initial value, which will add up to the total fitness value.

When the score of the skill drops below 5 the team's punctuation is drastically penalized, as the objective is to get the best team, with the best team members and it is necessary to discard the worst teams somehow.

$$f(c_{ij}) = \begin{cases} 1, & c_j \geq 9 \\ 0.7, & c_j = 8 \\ 0.6, & c_j = 7 \\ 0.5, & c_j \geq 5 \\ 0.1, & c_j < 5 \end{cases}$$

Equation 2 - Regular Fitness

Following the Fitness Equation, once the "regular" capabilities have been evaluated, it is time to define the function which is in charge of evaluating the Leadership of the team members, explain this decision of assessing this capability apart and justifying the values given to the function.

As it was previously explained, leadership is a desirable and positive capability on a worker, but it is also dangerous if several workers with high leadership happen to be in the same team it could lead to discussion among the most powerful members and lead to lowering the productivity of the whole team.

In order to avoid situations in which the general Leadership capability is very high, and the following function has been defined so the teams with high average on leadership, are penalized.

In Equation 3 formula, the threshold which has been set to start penalizing the teams activates the trigger when the average value of Leadership, considering all the members of the team has a value greater than seven.

That specific value has been chosen to establish an upper boundary that no team should pass (regarding leadership), because it is likely that if the average leadership of the team is 7, there are more than one leader in the team, and that is not a desirable situation.

In case the penalization happens, the value of the Fitness evaluation will be reduced by subtracting the sum of the different values of leadership of the team, divided by two, so the punishment will be proportional to the Leadership proportion and significant enough to make a difference with the other teams.

In case the threshold is not reached, the value of the function will be 0, so it will not affect the Fitness value at all, as this function has been designed to penalize the excess of Leadership.

The following equation summarizes the previous definitions of the Leadership evaluation displaying the two possibilities depending on the values obtained.

$$f(l_i) = \begin{cases} \dfrac{\sum_{n=1}^{i}(l_i)}{2}, & \dfrac{\sum_{n=1}^{i}(l)}{i} > 7 \\ 0, & \dfrac{\sum_{n=1}^{i}(l_i)}{i} \leq 7 \end{cases}$$

Equation 3 - Leadership evaluation

This evaluation could've not been done in the main equation part by just giving different weights to the leadership penalizing high values, because the leadership of each worker on their own is not important, the important is the combination of leadership values along the team and avoiding two leaders in one group.

Even though it has been said that leadership is a special case and has a special method to evaluate it, it is mainly used to punish undesirable team configurations, and I considered that it is necessary to include also the Leadership in the Regular Fitness function evaluation because, at the end, the purpose is making optimal teams, and although Leadership has to be evaluated separately, the punctuation and weight that it contributes to the evaluation of the team is also important and must be taken into account.

Operation Mode 2

In the second operation model, what we want is to generate the best team available like in the previous case, but highlighting one (or some) of the capabilities so we make sure that the team stands out in this particular capability.

This second mode is especially useful for projects which are focused in specific fields and some capabilities are essential for the proper development of the project. For example, if the project is related to the development of a web page or an application, creativity is surely a key factor for that project.

This adapted fitness function is the same as in the basic operation mode, and adds a new component at the end which is $g(e_i)$, a function which rewards the team configuration that have high values for the highlighted capability, giving them advantage at the time of being chosen for reproduction.

$$F = \sum_{n=1}^{j} \left(\frac{\sum_{n=1}^{i} \left(c_{ij} * w(c_{ij}) \right)}{i} \right) - f(t_i) + g(e_i)$$

Equation 4- Fitness Function 2

In the equation below we can see that the function has a similar behavior to $f(t_i)$ but just in the opposite way. If approximately the mean of values for that capability is higher than 7, that group will be rewarded with an addition of some more points in the fitness evaluation which will depend on their scores for that capability.

In case the team does not stand up for that capability, this function will have a 0 value, allowing the teams rewarded being picked easily because of their extra points.

$$g(e_i) = \begin{cases} \frac{\sum_{n=1}^{i}(e_i)}{2}, & \frac{\sum_{n=1}^{i}(e_i)}{i} > 7 \\ 0, & \frac{\sum_{n=1}^{i}(e_i)}{i} \leq 7 \end{cases}$$

Equation 5 - Extreme fitness

As it happened with the case of the Leadership evaluation, this Extreme fitness evaluation Works independently from the main equation, it is an extra feature, so the use of this second mode of operation for a particular capability does not exempt the highlighted capability from being evaluated in the general formula for Regular Fitness.

Operation Mode 3

This third operation mode, allows the user to determine which will be the methodology the team is going to be following through the development of the proposed project and it is an adaptation of the second mode of operation.

This operation mode can be useful in situations in a department in the company follows an specific methodology for software development and wants to know which team will be more suitable for the guidelines and needs of the methodology.

For this particular project, depending on the methodology chosen by the user, there will be a highlighted capability which will be the most representative for the methodology, or the one which is more useful at the time of developing following the methodology standards.

The classification assigning a Capability to methodology is defined in the State of the art section “Conclusions on methodologies".

Once each available methodology has their key capability assigned, the way to promote will be the same as in the second operation mode, adding a new function which will reward the teams with high scores for the chosen capability.

11.3.4. Selection

Once all the different individuals of the population have been evaluated and punctuated according to the corresponding fitness function and its weights, it is time to select which two individuals are going to be selected for the reproductive process.

The selection process implemented follows the philosophy of the tournament selection method; this method is based on randomly choosing some individuals from the population and then selecting the fittest ones from that selection for reproduction.

For the selection process, the given population is distributed randomly in two equal halves; in case the population number is uneven, the first half will allocate that remaining individual. Once the population is divided in two, the fittest team from each half will be selected for Crossover; as there are no repeated teams in the population; there is no chance of crossing to equal teams.

By doing the selection process this way, creating two different groups of individuals each time, we give the opportunity to some individuals which are unlikely to be chosen

to match with a fit individual, whose result could be useless, interesting or even beneficial; but always giving a greater probability of being chosen to the fittest individual, as the purpose of this system is always to obtain the best possible result.

11.3.5. Crossover

After the selection of the fittest team from each half of the population, the next step is the crossover of those two individuals with the aim of generating two offspring which could or could not be fitter than the parents who created them.

From this Crossover stage, two offspring will be obtained and the crossover modality will be multipoint. The way to select the crossover points will be doing randomly, to keep with the same dynamics as when creating the initial population.

In order to generate the first offspring some random genes from the first parent will be chosen to become part of the first offspring (with a maximum of four), then, the remaining genes left in the first offspring chromosome will be fulfilled with random genes from the second parents until the chromosome is complete.

The genes from the first parent which have not been chosen to be part of the first offspring will be part of the second offspring, which will be completed with the genes remaining in the second parent.

In order to do this distribution of genes, several verifications have to be made in order to assure that there is no possibility of creating a teams with two repeated genes, because that would imply that a team had twice the same worker in it, and that is not a feasible solution for the proposed problem.

After including the genes from each parent in the corresponding offspring, and to follow the ordination criteria, the genes will be reorganized and ordered from low to high so as to easily detect and avoid repeated team configurations

The following figure shows an example of the crossover schema designed for this project with the two selected parents and their resulting children.

Figure 41 - Crossover

The individuals situated on the left side are the parents; the genes selected to generate the first offspring are colored in red, while the genes chosen for the second child are colored in blue. On the left side of the picture, the two offspring generated from the crossover of the two parents can be seen (red and blue children).

11.3.6. Mutation

Once both offspring have been generated by the selected parents, as it happens in genetics, there is a small probability that a gene mutates. For this project, the mutation probability has been set to 0.1, so each time a child is evaluated, there is a ten percent chance that one gene from that chromosome may change to a different one.

In case the chromosome is selected for mutation, a random gene will be selected for mutation, and it will be replaced with a random gene from the existing ones (there cannot be a worker which does not exist). As in previous cases, it will be necessary to validate the incoming gene and checking that it does not already exists in the chromosome; in case the gene is already present in the chromosome, another random gene will be chosen.

The following figure shows an example of the mutation process:

Figure 42 – Mutation

It can be observed that, the gene colored in orange (37) is the gene which has been selected to be mutated, and in the right side of the picture the new value for the mutated gene (23) can be seen.

11.3.7. Add to population

Finally, after the two offspring have been generated and mutated (or not), it is time to include these two new individuals into the current population.

There are many diverse ways of introducing new members into the population; it is possible to create a new population from the new children, to use the current and add the children but removing the parents, to remove the less fit and add instead the offspring…

In this project this step is made simple; the new individuals will be added to the current population without removing any individual from the previous population.

11.3.8. Termination Condition

To finish, there is the Termination Condition which appears in the GA diagram; depending on the type of problem the ending condition could be related to resources, or to reaching a value considered optimal enough.

The problem with the Termination based on reaching a defined value, is that such condition may never be reached depending on the randomness of the Genetic Algorithm and the program may end up stuck without finding a solution.

For this reason, in this case the Termination Condition has been based on a fixed number of iterations which hopefully will be enough to lead the program to a optimal (or almost optimal) solution for a given problem.

Chapter 12 | Case study: Obtaining the best team

In this chapter, a case study using this genetic algorithm-based system will be presented, starting with a prior assessment of the proposed problem and the data to be evaluated, the series of experiments done and an analysis and justification of the results obtained from the experimentation phase.

12.1. Prior assessment

For this case study, the operation mode which is going to be used for experimentation is the first one, in which the system provides the user with the best team configuration without highlighting any particular capability.

The information about the different workers and their capabilities has been extracted from students of Computer Science in the university Carlos III of Madrid. These evaluations of the capabilities were obtained through the completion of a 360º review after the completion of "Software Engineering" course and it was provided by my tutors. The data sampled for this study case can be consulted in Appendix V.

For this case study a sample of 50 people will be used, each of these users will have 9 different capabilities and the best team consisting of 5 workers must be configured using the first operation mode described in the Selection section, without highlighting any capability or using a specific methodology.

12.2. Experimentation and analysis of the results

In order to evaluate this study case, it was necessary to carry out several experiments with the proper configuration of the system in order to evaluate the proposed problem.

The program was configured to use as an input the file containing the 50 workers with their 9 capabilities evaluated each, with a team size of 5 and a maximum number of iterations of 1500.

After the several execution of this test case in the designed program, the best team configuration was selected for the evaluation, this means, the team with the highest fitness value along several executions of the program was chosen.

The chosen team had a value obtained in the fitness evaluation of 63.62 and it was configured with the workers whose ID was:

- Worker 4
- Worker 6
- Worker 10
- Worker 16
- Worker 18

In the following table, the different scores for each of the team members for every different capability are provided for a better understanding of the chosen team.

	Worker 4	Worker 6	Worker 10	Worker 16	Worker 19
Teamwork	8	7	8	9	6
Leadership	7	7	9	5	7
Commitment	9	6	10	8	9
Planning	9	9	6	7	8
Creativity	7	10	9	9	10
Social Skills	9	7	9	5	9
Negotiation Skills	10	9	10	8	9
Programming Skills	9	9	10	10	5
Analysis and Design Skils	10	10	10	10	6

Table 78 - Team configuration

It can be observed, that in general the scores of the workers in the different attributes are quite acceptable, and they have a reasonable high average value in all the competences.

Another important factor to look at is the leadership; it is easy to see that Worker 10 has high leadership skills, and that the rest of the team members have more moderated values for that competence in order to compensate the leadership of that member.

In general terms, all capabilities are highly covered by one or more team members and each team members is an expert on several areas, compensating their lacks on some

aspects as for example, Worker 19, with lower technical skills, compensates with great creativity, social and negotiation skills.

In order to validate the accuracy and precision of these results obtained from the execution of the developed program, the opinion of various experts was requested. The team of experts was composed of 5 people, three of them belonging to the University Carlos III of Madrid and two of them belonging to ADIC ("Asociación para el Desarrollo para la Ingeniería del Conocimiento", Madrid).

Each of the experts were asked to provide five possible team configurations which they consider optimal given the same 50 workers used in the execution of the program (repeating workers in different teams, if desired), obtaining a total of 25 different team configurations from the experts' sample.

Comparing the results obtained with the samples provided by the experts, it was very complicated that the exact team configuration was also proposed in the sample given the small amount of samples and that was the case, there were no configuration in the experts' choices which matched the proposed configuration at 100%.

On the other hand, positive outcome was extracted from this experiment, as even if the team configuration itself hadn't been selected, the team members contained in the best team configuration, were one of the most chosen by the experts even though they used different configurations.

The percentage in which of the user in the selected team appears in some team of the ones designed by the experts are the following:

- Worker 4 ➔ 52%
- Worker 6 ➔ 28%
- Worker 10 ➔ 60%
- Worker 16 ➔ 44%
- Worker 18 ➔ 32%

This tells us, that some of the best users that the system considered were also considered by the experts and some of them in a very high percentage as Worker 10 and Worker 16.

Going back to the team configurations, it was difficult that the exact team configuration were selected by the consulted experts, but it was considered important in which percentage team configuration matched, even if it wasn't the whole team configuration, but some of their members.

The following pie chart shows the distribution of occurrences of the proposed team members in the experts' teams. It can be observed, than in 8% of the groups none of the workers in the designed team appears, in 16% of the groups there is only one member that matches with the selected ones, in 52% of the groups there are two members which are also in the designed team and in 24% of the teams there are three members which are the same as in the team designed by the algorithm.

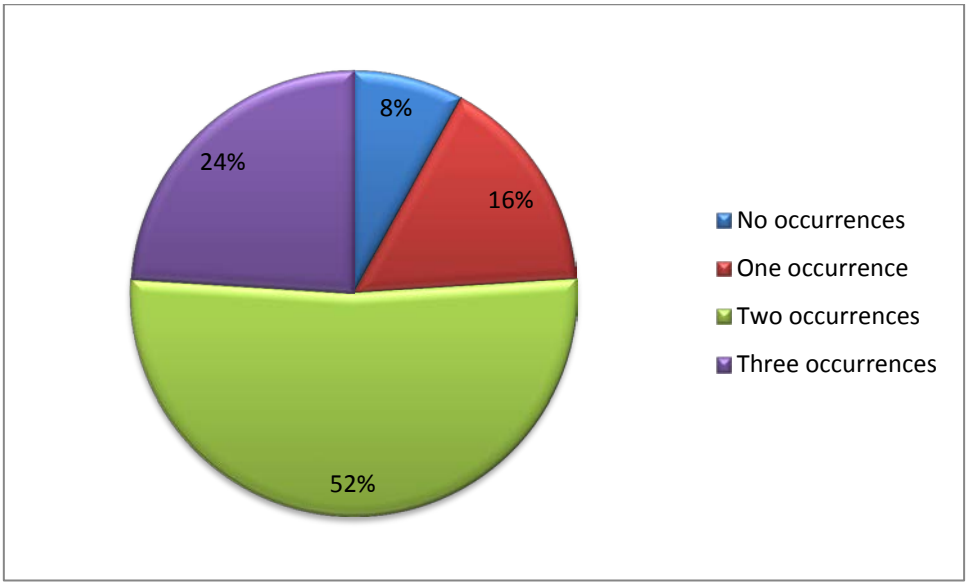


Table 79 - Percentage of occurrences

From this data we can deduce that:

- **92%** of the teams have at least one member from the designed team
- **72%** of the teams have at least two members from the designed team
- **24 %** of the teams have three members from the designed team.

These results are quite encouraging if the remember that teams are only configured with 5 workers and 72% have at least two matching team members, which is almost half of the team members.

For future experimentation, it would be interesting to consider different sampling sizes, requiring more samples from the experts and changing team sizes, in order to see how the number of workers in a team change the results and ponder on possible improvements on the algorithm in order to obtain more accurate results.

Chapter 13 | User Interface Design

In this chapter, the reader will be provided with some mockups images of how the system should look like if the User Interface had been developed. These figures are only an approximation of the final system Interface, but they facilitate to understand the way the system works, the options end users have to interact with and the distribution of the different modules and their functionalities.

In the first figure we can see the register screen, in which the user has to type a username and an e-mail, both of them have to be unique in the system's database, otherwise, the user will have to choose a different username.

For security reasons, the user will have to type his password twice so as to assure that it has not been misspelled.

Figure 43 - User Interface: Register

Once the user has been successfully registered into the system and added to the database, he will be asked to Log In with the previously defined credentials. The required fields to fulfill by the user are the username and the password. If the username matches the password, the user will have full access to all the system's functionalities.

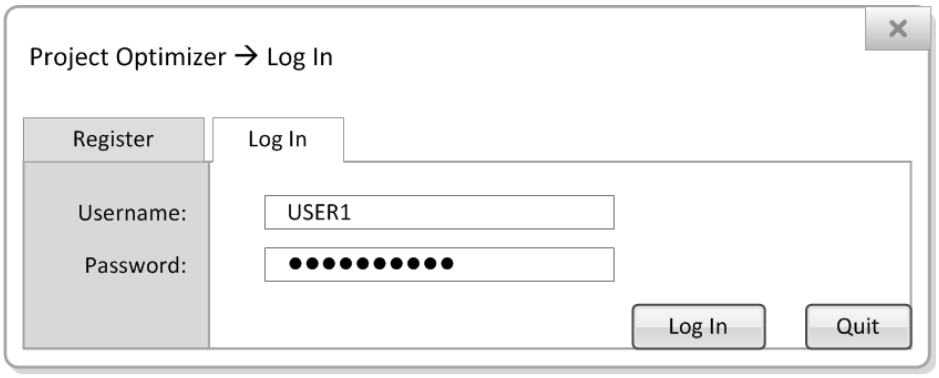


Figure 44 - User Interface: Log In

The first screen the user is going to find when he logs into the system is the "Create Project Screen" in which he can give an identifier to the new project, as well as defining its size. Once the user has properly defined the project's attributes he will click on "Save Project" in order to store the new project in the database and being able to work with it.

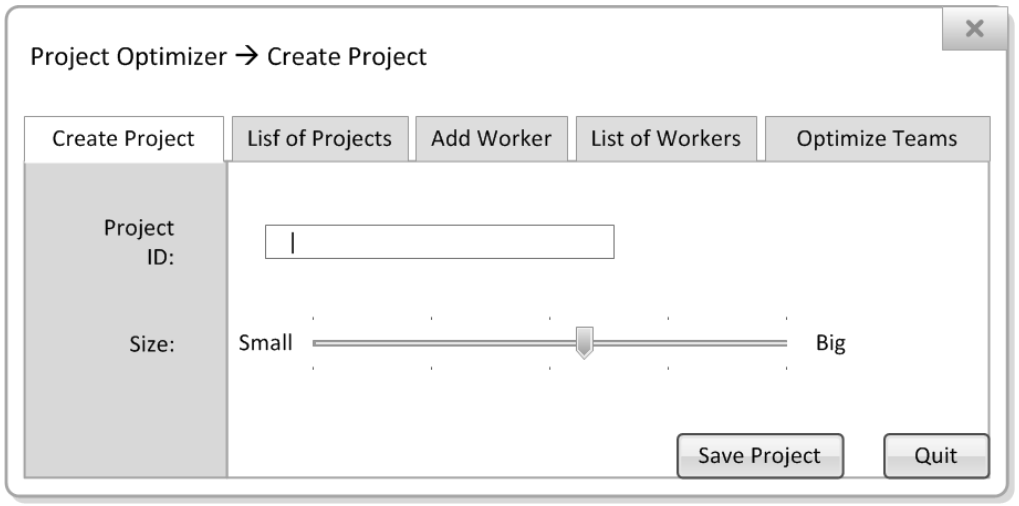


Figure 45 - User Interface: Create Project

In case the worker has previously used the program before and created some projects, these will be available on the "List of Projects" tab, which will allow the users to Open an existing project to work with it, or to delete it in case is not needed anymore.

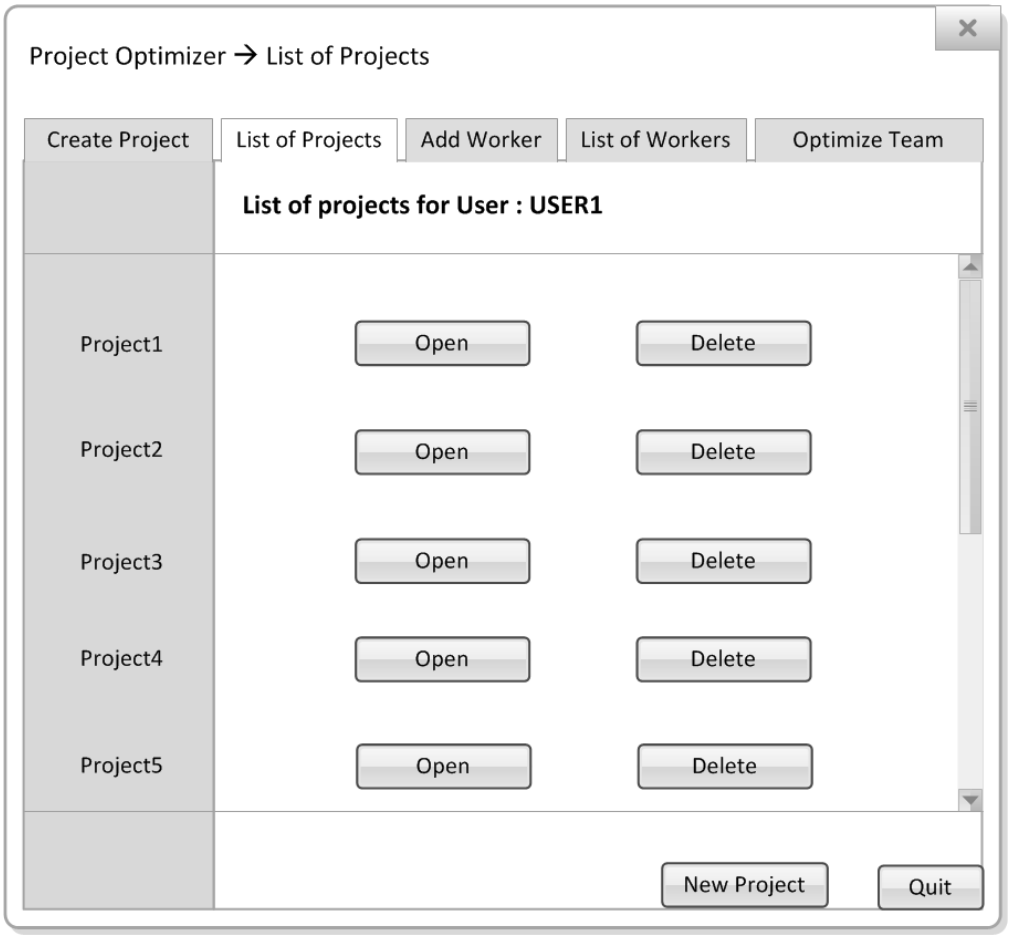


Figure 46 - User Interface: List of Projects

The next step for the team optimization, is adding potential workers to the list of workers of the project, in order to do it, the user will have to click on the "Add Worker" Tab, and define each of the attributes of the new worker, as well as an Identifier so as to distinguish him from the other workers.

Once the user is satisfied with the defined parameters, he will have to click on the "Add Worker" button in order to add him to the list of workers.

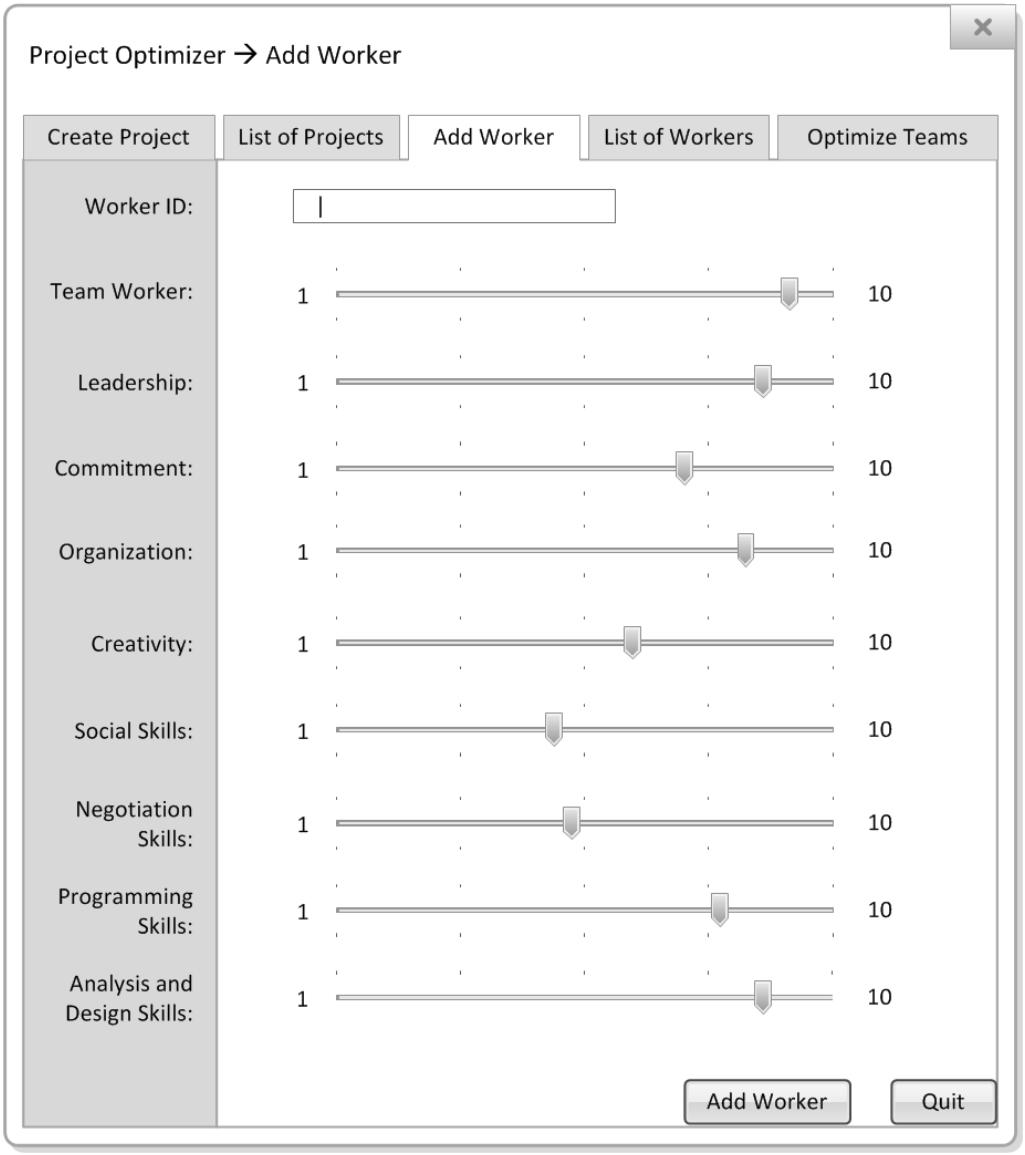


Figure 47 - User Interface: Add Worker

When the first user has been added to the project list, the "List of Workers" tab will be available for the user; in this list the user can consult the list of available workers for the current project, have a look at their characteristics, modify their attributes or delete them if they are no longer considered for the project.

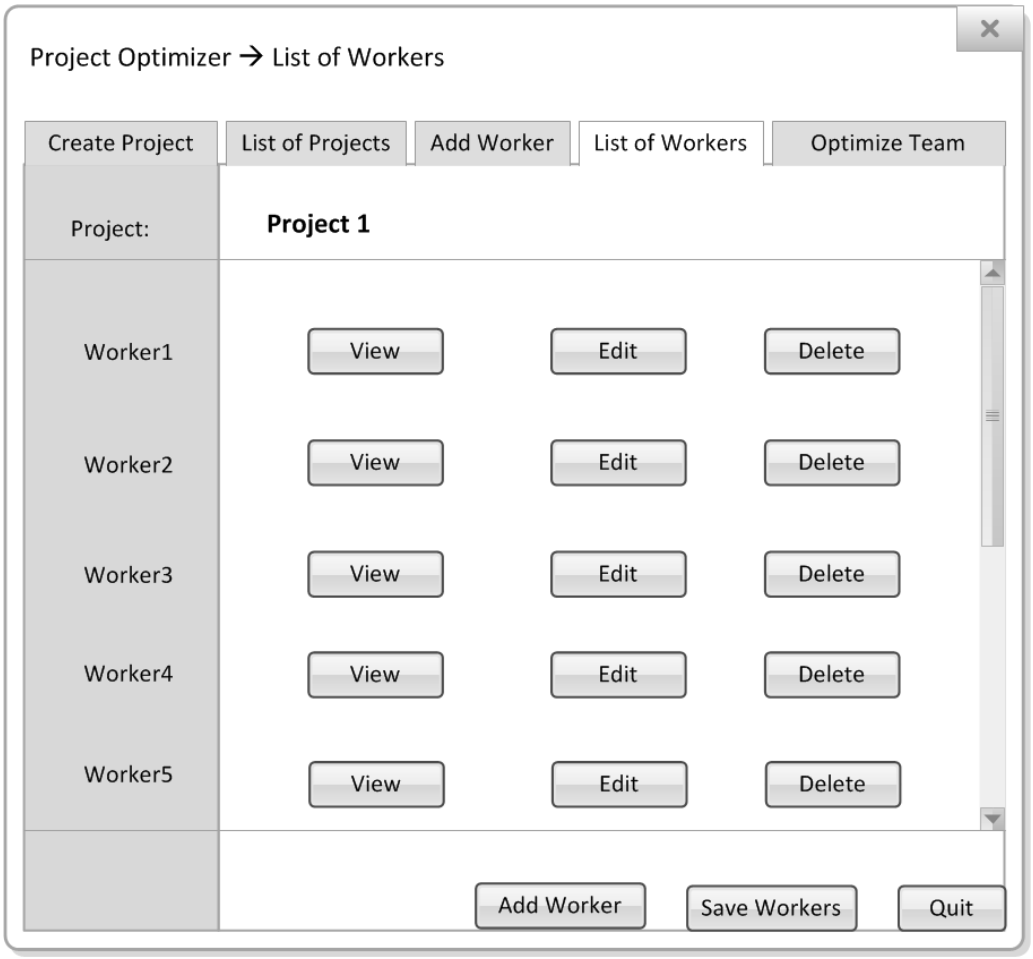


Figure 48 - User Interface: List of workers

Once the project has been created and the users have been added, the user can go to the "Optimize Team" tab, which will allow him to select an optimization mode from a set of possibilities. Depending on the selection of the user, the results obtained will vary as the objectives pursued for each operation mode are slightly different.

When the optimization mode has been chosen, the user may click on the "Optimize" button and the results will be displayed on screen; the user can now decide if he wants to close the results window or export the results of the optimization in order to store them locally.

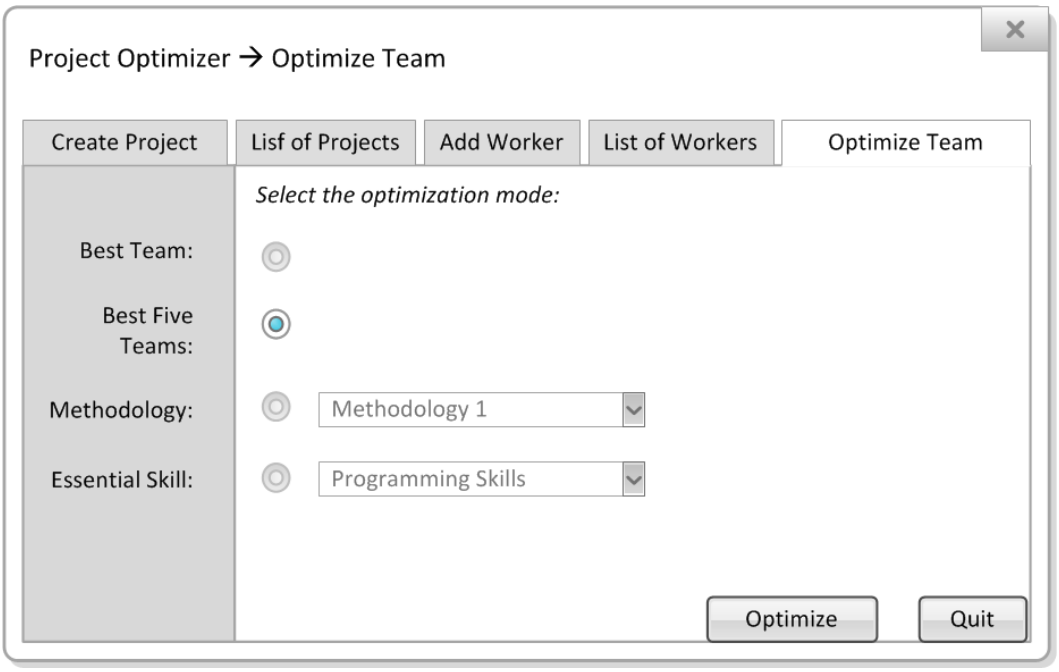


Figure 49 - User Interface: Optimize Team

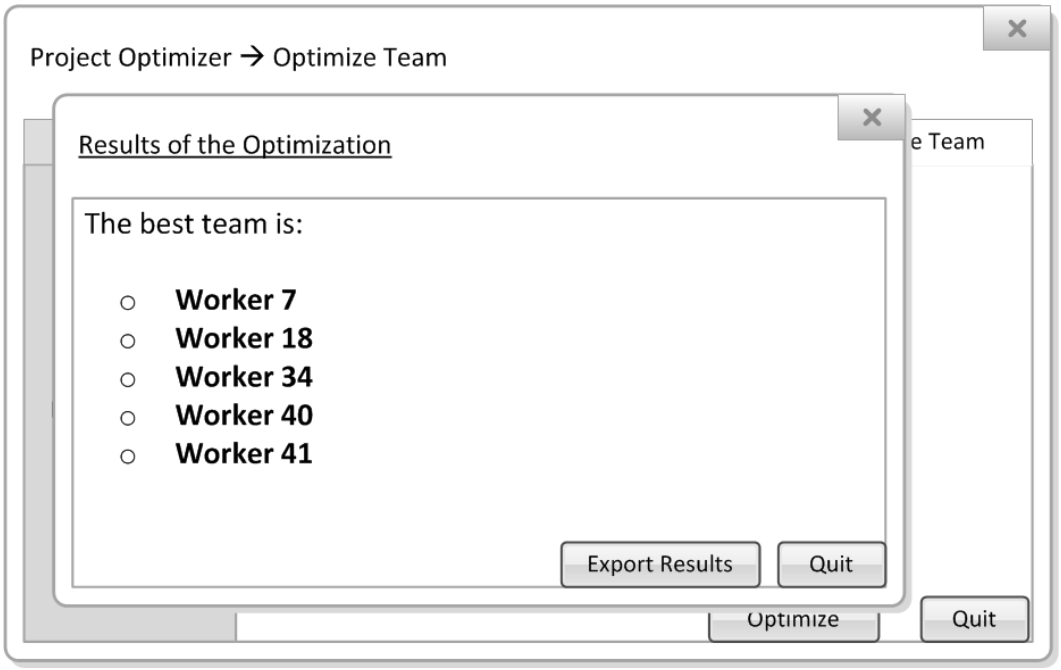


Figure 50 - User Interface: Results Dialog

When the user has finished using the application, he must click on the exit button at the right corner at the top of the screen in order to log out and exit the program.

PART VI:

CONCLUSIONS

Chapter 14 | Conclusions and future lines of work

Throughout this chapter, the reader is provided with information on the different difficulties encountered along the development and the documentation phases of these projects, as well as personal conclusions and thoughts and finally, some possible future lines of work to follow in case this project continued to be studied and developed.

14.1. Difficulties encountered

Along the different stages involved in the development of this project, several difficulties were encountered, luckily it was possible to deal with them and learn from it for similar future situations. The main challenges found during the development of this project were:

- **Limiting the scope of the problem**: As working teams and human resources are such wide topics, it was difficult at the beginning limiting the domain of the proposed problem to some specific study fields. Leaving behind some interesting analysis topics, and restrict the functionalities of the system to something realistic according to the available time and resources was hard at the beginning of the development and took more time and planning than it was initially estimated. The positive outcome of this obstacle is the desire to continue with this development in the future in order to wide and explore all the different possibilities which had to be left behind because of the mentioned constraints.

- **Understanding of Genetic Algorithms**: One of the main difficulties found in this development, was the proper understanding of the Genetic Algorithms and their inner functionalities. The main difficulty was related to the proper understanding of the fitness function and its importance of its correct design.

- **Definition of the solution**: This problem is tightly related to the previous one; at the time of defining the fitness function to be used in the proposed solution, a long time was required to reach to a suitable and logical design of the equation which will be right for the proposed problem and would provide accurate results. A lot of effort was required in order to define the function and the different weights corresponding to each of the different capabilities, as well as the definition of the other stages of the GA which even so, were easier to understand and develop than the fitness function.

- **Processing of the results**: Finally the last trouble encountered was at the time of evaluating the results obtained from the Genetic Algorithm system. It was necessary to study the way these results were going to be presented and it was difficult to interpret correctly the results and explaining the conclusions extracted from them.

Despite the challenges mentioned above, all of them were overcome and the project was completed successfully.

14.2. Conclusions

Once all the difficulties and obstacles have been discussed, it is possible to relate the general conclusions and general feelings about the project now its development is concluded.

With this project, now it is possible to generate and optimize software development working teams from a given set of workers with a project of certain size. It also allows emphasizing some features which may be found desirable and can orient the team distribution to a specific development methodology.

Even though obstacles and doubts were found along the way of this development, they have been successfully resolved and the purpose of the project was able to be fulfilled including all the desired functionalities and providing satisfactory results.

The overall impression on the project after it has been developed and completed is very positive; the objectives set at the beginning of the project have been reached and have been properly carried out and there is a wide range of possibilities of use for this system in the real world.

This project can be helpful in real companies where development teams have to be created and distributed and the person in charge can be advised and helped by this system, making decision making easier, saving time and economic resources.

Additionally, this project could be easily broadened to several areas of work, not only software development, thus extending its possibilities of use for different kind of people and companies, making it a useful and desirable tool for teams management.

Summarizing, the level of satisfaction after the development of this project is high, and the potential of expansion and usage in the real world are very encouraging.

14.3. Future lines of work

Because of the time limitations imposed for the development of this Bachelor Thesis, it was not possible to deepen as much as it was wanted in the subject of study and not all the desired functionalities could be analyzed and developed.

The following list provides a set of functionalities and upgrades which would be desirable and would add value to the project in case it were to be continued in the future.

- **Development of a User Interface**: because of time constraints, it was not possible to design a Graphic User Interface for the program, it would be highly desirable to implement the designs provided in Chapter 13 and integrate them with the system to make it usable and friendly to future users in case the project were developed

- **Improvement of the fitness function:** in future works, it would be highly interesting to have more time to experiment with the fitness function, trying different ways of evaluating the individuals and obtain more accurate results than the ones obtained for this project.

- **Considering new capabilities**: Additionally, if new capabilities were added to the workers, these could provide a whole new set of different analysis choices. For example, if the relationships between the employees were scored and considered, the possibility of configuring teams in which those workers would not be together arises.

- **Increase the operating modes**: Another improvement which could be applied to the system is increasing the operating modes available. Currently, from a set of several workers, it is only possible to select the best team configured with that worker. It would be a desirable enhancement if the system were able to distribute the available workers into different teams, which could be balanced teams so every team would be similar, or unbalanced teams, in which the best team would be generated, with the remaining individuals the second team would be created and so on. Another possible operation mode is not only defining the team in function of the project and the methodology used, but to simultaneously choose the methodology and the team which would be more suitable for a particular project.

- **Complement the Genetic Algorithm with other techniques**: Another suggestion for future works would be the study of techniques which are compatible or complementary to GAs such as neural networks, in order to experiment if a supporting technology would help to improve the results of the Genetic Algorithm by itself.

- **Expand the field of study**: Finally, another branch in which this project could be further developed is the expansion of the field of study, not only considering software development working teams, but any kind of team, since marketing teams or project teams at university.

PART VII:

APPENDICES

Appendix I. Acronyms

- **UP**: Unified Process
- **RUP**: Rational Unified Process
- **LD**: Lean Development
- **FDD**: Feature Driven Development
- **RAD**: Rapid Application Development
- **DSDM**: Dynamic Systems Development Model
- **XP**: Extreme Programming
- **GA**: Genetic Algorithm
- **DNA**: Deoxyribonucleic acid
- **UML**: Unified Modeling Language
- **VAT**: Value Added Tax

Appendix II. Planning

In this appendix, the estimated and actual planning of the project development will be provided, as well as the budget distribution needed for the development of the project.

In Figure 55, the initial planning estimation for the development of this project is presented. The development started the 1st of May and established the end the day of the final presentation.

The initial planning proposed a model in which started with the development of the genetic algorithm and was able of carrying out development and documentation tasks at the same time since middle May.

The way of chaining tasks was very similar to waterfall methodology, once a task is finished and closed, it moves to the next one and so on until the last task,

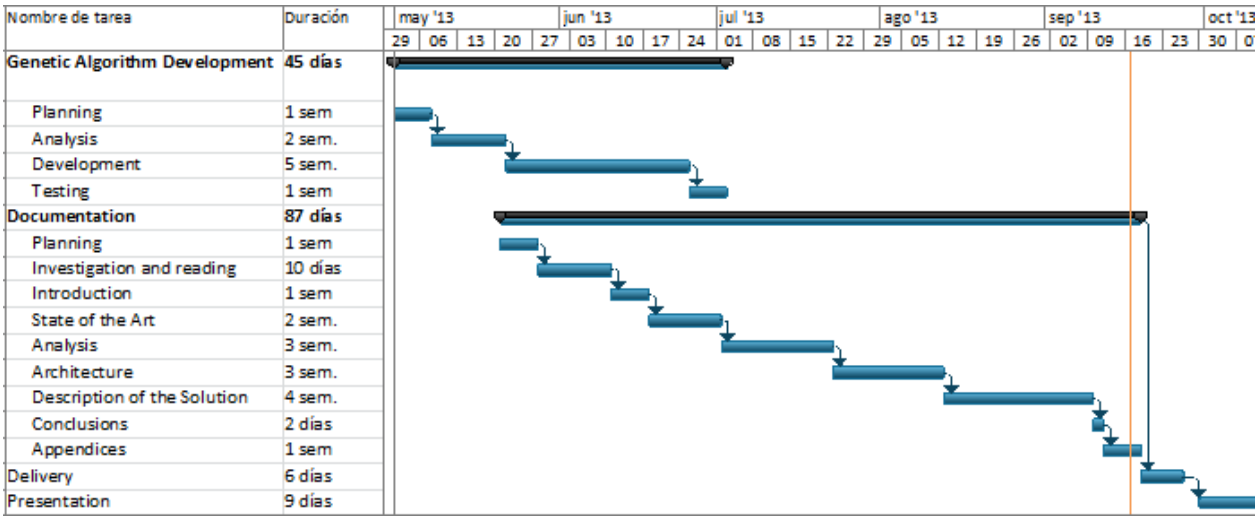


Figure 51 - Initial Planning

In Figure 56 the actual planning which was followed during the development and documentation of the project is shown. It can be seen that the development starts and end on the same dates, but the way of managing and completing the different tasks is a little different.

To start with, development tasks were estimated to take 45 days and at last only 35 days were needed, the same happened with documentation, 87 days were estimated and 72 were used at the end.

The main difference between the planning and the actual development is that documentation phase was not started until development phase was almost over, which reduced the time for documentation from the estimation.

There are also differences in the order in which documentation tasks have been completed and in the simultaneity of some of the tasks, but in general lines, the estimated planning did not deviate as much from the actual development and documentation time.

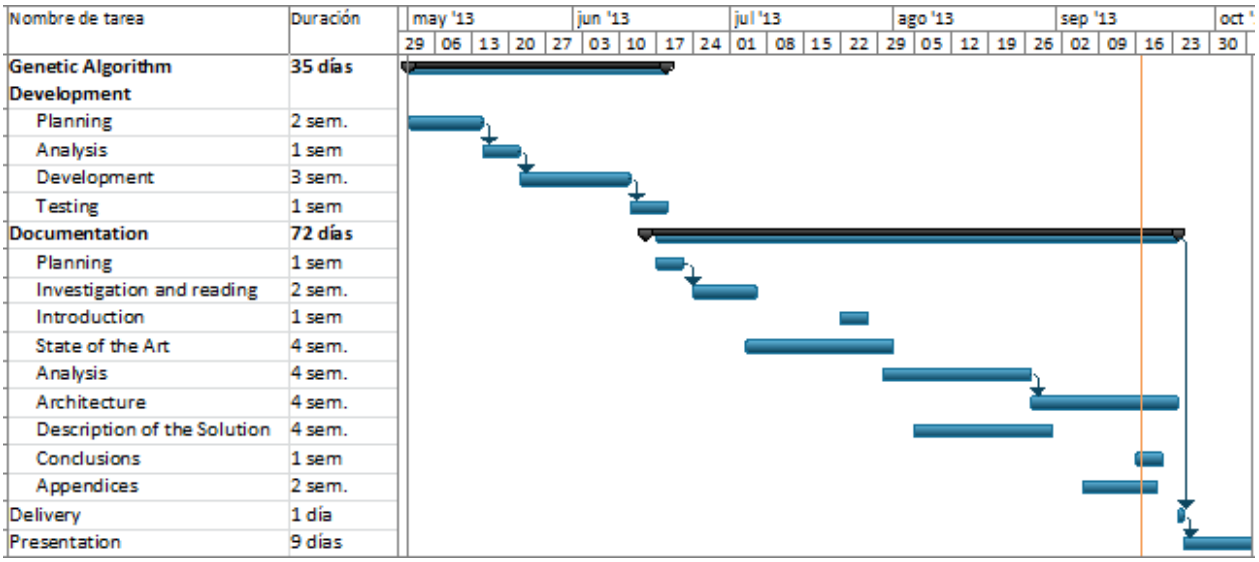


Figure 52 - Real Planning

Figure 55 displays the effort distribution (in days) for the development of the Genetic Algorithm program, and Figure 56 shows the effort dedicated to all the different documentation tasks.

From the development distribution, it was expected that the heavier tasks would be the actual development of the GA program and in the documentation distribution it can be observed that the main four parts of this document (state of the art, analysis, architecture and description of the solution) were the parts which required most of the time, and also are the most extensive sections in the document.

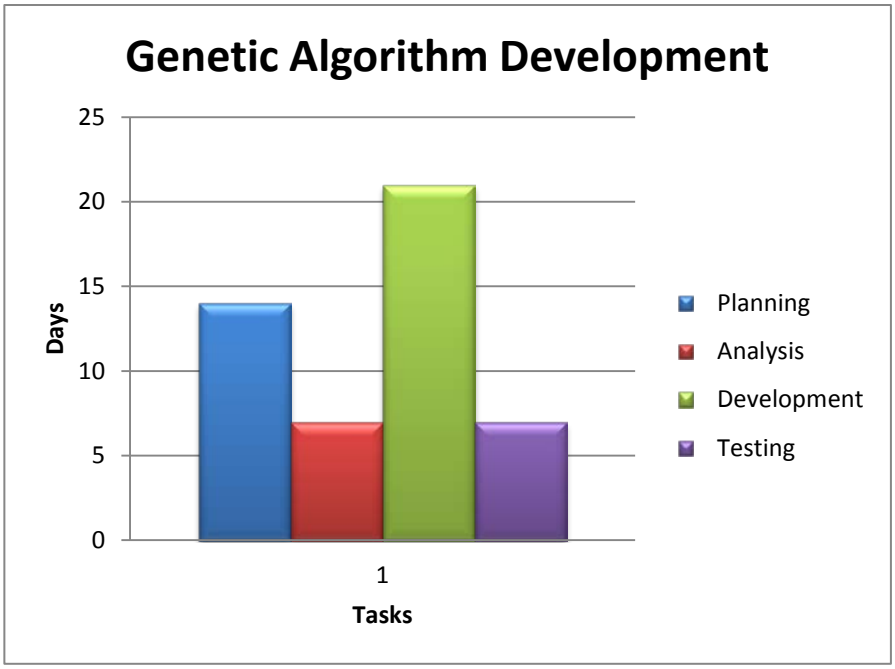


Figure 53 - GA development effort distribution

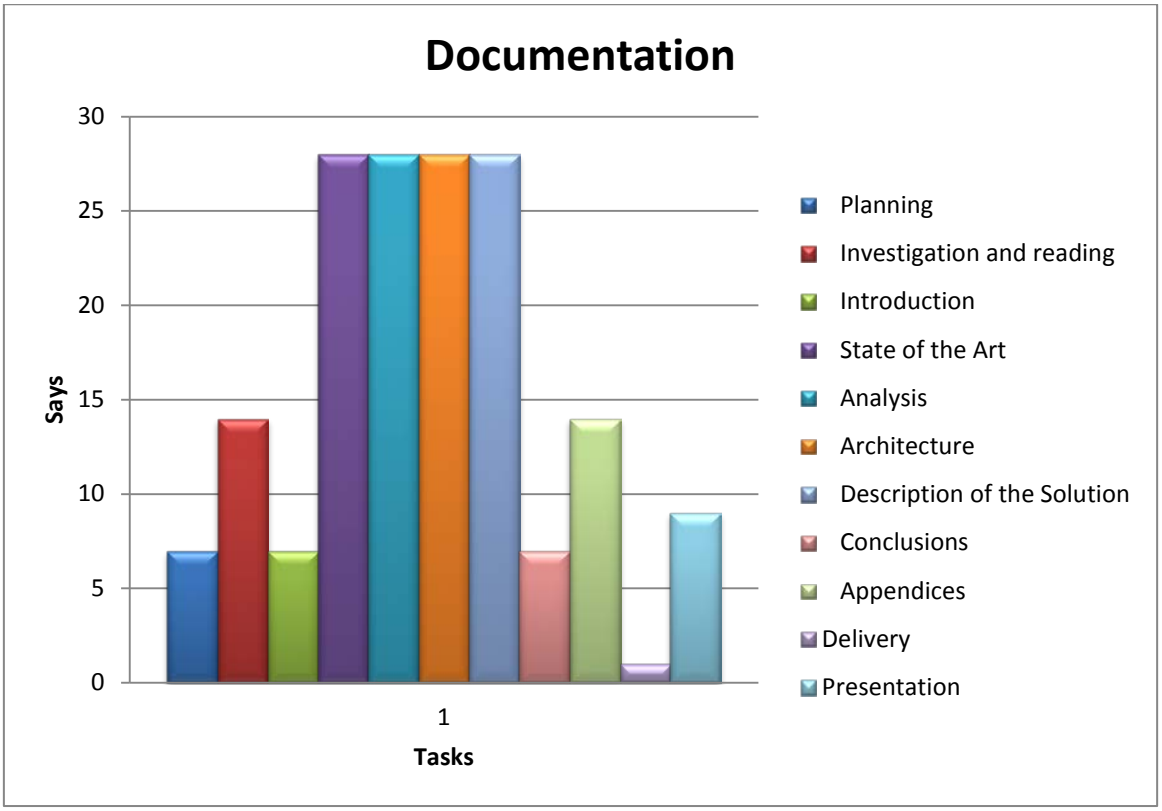


Figure 54 - Documentation distribution

Appendix III. Budgeting

In this appendix, the associated costs and the budget required for the development of the project will be defined and explained in detail.

In the following table, the salary which would be paid to the worker in charge of the development of the project is displayed. It provides the salary per hour and per week of the worker, as well as the total salary to be paid for a work done in 16 weeks.

Salary per hour	Hours/week	Salary/week	Total (16 weeks)
12 €/hour	25 hours/weak	300€	**4800 €**

Table 80 - Worker's salary

Table 81 lists the different amortized costs, which are the ones derived from the working place rental, equipment, software licensing and so on.

Issue	Cost
Office rental	400 €
Computer rental	100 €
Microsoft Office 2013 rental	40 €
Microsoft Visio 2013 rental	160 €
Microsoft Project 2013 rental	280 €
Total	**980** €

Table 81 - Amortized costs

The next table shows other expenses which are not included in the previous charts which include the transport costs and the spent office supplies.

Issue	Cost
Office supplies	40 €
Transport	160 €
Total	**200** €

Table 82 - Other expenses

Finally, this last table shows the different costs explained before together, and provide the total costs of the project with and without VAT (Value Added Tax), which in Spain is the 21% of the total cost of the project.

Issue	Cost
Salaries	4800 €
Amortised costs	980 €
Other expenses	200 €
VAT	21% (1255 €)
Total without VAT	**5980 €**
Total with VAT	**7235.8 €**

Table 83 - Total cost of the project

Appendix IV. Software and tools used

This section contains a list of all the different software tools which were used in the development and documentation stages of the process:

- **Microsoft Office 2010**:
    - **Word**: Used for writing the Bachelor Thesis's report.
    - **Excel**: Used for analyzing the data extracted from the experiments and comparing it.
    - **Power Point**: Used for preparing and designing the presentation of the Bachelor Thesis.
    - **Visio**: Used for the design of the UML diagrams, Use Case diagrams, Flow diagrams and more illustrations used in the report.
    - **Project**: Used for the planning and scheduling of the different stages and tasks of the problem.
- **Draw.io** (web): Used for the design of some of the diagrams and figures used in the report.
- **Eclipse Juno** (Java): Used for the development of the Genetic Algorithm-Based System and the study case experimentation.

Appendix V. Data sampled

	Teamwork	Leadership	Commitment	Planning	Creativity	Social Skills	Negotiation Skills	Programming Skills	Analysis and Design Skills
Worker 1	7	9	6	7	10	8	6	10	6
Worker 2	7	6	8	10	8	7	8	6	8
Worker 3	10	9	8	8	10	9	6	7	6
Worker 4	8	7	9	9	7	9	10	9	10
Worker 5	9	10	10	9	8	10	6	7	10
Worker 6	7	7	6	9	10	7	9	9	10
Worker 7	7	10	8	9	10	10	6	7	8
Worker 8	10	9	9	10	8	8	10	8	7
Worker 9	10	9	10	10	7	7	7	6	9
Worker 10	8	9	10	6	9	9	10	10	10
Worker 11	10	5	5	7	6	8	8	5	9
Worker 12	8	10	7	6	6	6	7	6	8
Worker 13	7	8	6	9	8	6	8	5	8
Worker 14	7	5	9	8	6	9	6	10	6
Worker 15	10	6	8	6	6	5	6	9	5
Worker 16	9	5	8	7	9	5	8	10	10
Worker 17	8	7	7	9	10	8	7	6	6
Worker 18	6	7	8	5	10	6	7	10	9
Worker 19	6	7	9	8	10	9	9	5	6
Worker 20	5	10	9	7	8	9	9	10	8
Worker 21	2	10	3	6	9	5	5	3	9
Worker 22	4	4	5	8	6	3	7	5	8
Worker 23	7	2	9	3	7	3	4	3	6
Worker 24	9	3	6	5	10	8	8	3	2
Worker 25	9	8	10	9	6	6	9	8	8
Worker 26	3	5	3	2	3	8	3	10	6
Worker 27	7	6	9	2	4	6	6	9	8
Worker 28	10	2	6	9	7	4	2	7	4
Worker 29	10	5	3	10	2	8	5	5	10
Worker 30	4	5	10	8	4	5	7	4	10
Worker 31	7	2	3	3	6	7	8	6	4
Worker 32	6	9	5	8	5	4	10	10	9
Worker 33	2	3	5	8	7	6	4	10	10
Worker 34	4	7	6	4	2	7	8	7	5
Worker 35	2	3	4	9	5	6	4	10	6

Table 84 – Data sampled part 1

	Teamwork	Leadership	Commitment	Planning	Creativity	Social Skills	Negotiation Skills	Programming Skills	Analysis and Design Skills
Worker 36	9	9	2	3	10	9	10	4	10
Worker 37	7	5	10	10	4	2	8	9	2
Worker 38	9	9	10	9	8	9	3	2	3
Worker 39	10	5	3	7	10	4	4	5	3
Worker 40	10	2	5	7	3	4	9	8	10
Worker 41	2	3	4	3	5	5	1	3	3
Worker 42	2	5	1	6	4	6	1	6	5
Worker 43	4	6	1	3	4	5	2	3	4
Worker 44	6	1	6	1	4	3	2	1	2
Worker 45	6	4	3	6	6	3	1	6	1
Worker 46	6	2	3	1	3	5	2	5	3
Worker 47	6	5	6	6	4	1	3	4	5
Worker 48	1	2	3	1	3	1	5	4	6
Worker 49	5	5	3	4	4	6	1	2	4
Worker 50	2	4	4	5	2	2	5	5	5

Table 85 - Data sampled part 2

Appendix VI. References

[1] A. Cockburn, Agile Software Development: The Cooperative Game, Addison Wesley, 2006.

[2] K. Meffert, "JAGP. Java Genetic Algorithms Package," [Online]. Available: http://jgap.sourceforge.net/.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, New York: Addison-Wesley Professional, 1998.

[4] L. D. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

[5] O. Hasançebi and . F. Erbatu, "Evaluation of crossover techniques in genetic algorithm based," *Computers & Structures,* vol. 78, no. 1-3, pp. 435-448, 2000.

[6] M. A. Awad, "A Comparison between Agile and Traditional Software Development Methodologies," 2005. [Online]. Available: http://pds10.egloos.com/pds/200808/13/85/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf.

[7] A. Neininger, . N. Lehmann-Willenbrock, A. Henschel and S. Kauffeld, "Effects of team and organizational commitment – A longitudinal study," *Journal of Vocational Behavior,* vol. 76, no. 3, pp. 567-579, 2010.

[8] N. B. Moe, T. Dingsøyr and T. Dingsøyr, "A teamwork model for understanding an agile team: A case study of a Scrum project," *Information and Software Technology,* vol. 52, no. 5, pp. 480-491, 2010.

[9] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology,* vol. 50, no. 9-10, pp. 833-859, 2008.

[10] C. Darwin, On the Origin of Species, Empire Books, 2010.

[11] J. Whalley, "Creativity and discipline quality management in software," *Microelectronics Reliability,* vol. 34, no. 12, p. 1963, 1994.

[12] M. Verkama and P. Heiskanen, "Comment on a decision support approach for negotiation: Software vs. methodology," *European Journal of Operational Research,* vol. 96, no. 1, pp. 202-204, 1997.

[13] I. McChesney and S. Gallagherb, "Communication and co-ordination practices in software engineering projects," *Information and Software Technology*, vol. 46, no. 7, pp. 473-489, 2004.

[14] M. Sutter and C. Strassmairc, "Communication, cooperation and collusion in team tournaments—An experimental study," *Games and Economic Behavior*, vol. 66, no. 1, pp. 506-525, 2009.

[15] M. Gómez, N. Juristo and S. T. Acuña, "How do personality, team processes and task characteristics relate to job satisfaction and software quality?," *Information and Software Technology*, vol. 51, no. 3, pp. 627-639, 2009.

[16] K. R. Laughery and K. R. Laughery Sr., "Human factors in software engineering: A review of the literature," *Journal of Systems and Software*, vol. 5, no. 1, pp. 3-14, 1985.

[17] O. Hazzan and I. Hadarb, "Why and how can human-related measures support software development processes?," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1248-1252, 2008.

[18] "Management Study Guide," 2013. [Online]. Available: http://www.managementstudyguide.com/understanding-team.htm.

[19] M. Gestal, D. Rivero, J. R. Rabuñal and J. Dorado, "Introducción a los Algoritmos Genéticos y la Programación Genética," 2010. [Online]. Available: http://www.galeon.com/dantethedestroyer/algoritmos.pdf.

[20] "Skills You Need. Helping you develop life skills.," 2013. [Online]. Available: http://www.skillsyouneed.com/.

[21] ESA Board for Software Standardisation and Control, "Guide to the user requirements definition phase," 1995. [Online]. Available: ftp://ftp.estec.esa.nl/pub/wm/anonymous/wme/bssc/PSS0502.pdf.

[22] ESA Board for Software Standardisation and Control, "Guide to the software requirements definition phase," 1995. [Online]. Available: ftp://ftp.estec.esa.nl/pub/wm/anonymous/wme/bssc/PSS0503.pdf.

[23] Oxfort University Press, "Oxfort Dictionaries," 2013. [Online]. Available: http://oxforddictionaries.com/es.

[24] D. Rodríguez, M. A. Sicilia, E. García and R. Harrison, "Empirical findings on team size and productivity in software development," *The Journal of Systems and*

Software, vol. 85, pp. 562-570, 2012.

[25] S. Faraj and L. Sproull, "Coordinating Expertise in Software Development Teams," *Management Science,* vol. 46, no. 12, pp. 1554-1568, 2000.

[26] F. Q. d. Silva, A. C. C. França, M. Suassuna, L. M. d. S. Mariz, I. Rossiley, R. C. d. Miranda, T. B. Gouveia, C. V. Monteiro, E. Lucena and E. S. Cardozo, "Team building criteria in software projects: A mix-method replicated study," *Information and Software Technology,* vol. 55, no. 7, pp. 1316-1340, 2013.